COPY

Naval Research Laboratory

Washington, DC 20375-5000

AD-A205 461

NRL Memorandum Report 6377

A User's Guide to Raytrace

DR. MICHAEL H. REILLY

Ionospheric Effects Branch
E. O. Hulburt Center for Space Research
Space Science Division

DR. ERIC L. STROBEL

Interferometrics, Inc.
8150 Leesburg Pike, Suite 1400
Vienna, VA 22180

February 8, 1989

DTIC
ELECTE
8 MAR 1989
S D
E

Approved for public release; distribution unlimited.

89

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Memorandum Report 6377	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Research Laboratory 6b OFFICE SYMBOL (If applicable) Code 4180	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000	7b ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
A User's Guide to Raytrace

12. PERSONAL AUTHOR(S)
Reilly, M.H. and Strobel,* E.L.

13a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Memorandum	FROM 10/87 TO 4/88	1989 February 8	295

16 SUPPLEMENTARY NOTATION
*Interferometrics, Inc. 8150 Leesburg Pike, Suite 1400
Vienra, VA 22180

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Ionosphere, Raytrace, Raytracing, Simulation, Three-dimensional

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The operation of the RAYTRACE (V.4.3) software for ionospheric raytracing is explained. First, the installation and setup of the program are discussed. Instructions for routine operation are given, followed by several tutorial examples. Some suggestions for regular usage are given. The final section and the appendices contain useful reference material, including the source code listing.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Michael H. Reilly	22b TELEPHONE (Include Area Code) (202) 767-2891 22c OFFICE SYMBOL Code 4180.2

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

CONTENTS

Accession For
NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

A USER'S GUIDE TO RAYTRACE

PART ONE:
INTRODUCTION

Manuscript approved April 22, 1988.

OVERVIEW

This manual is a user's guide to the operation of the program RAYTRACE (Version 4.3) and also covers related topics. As a user's guide, this document is meant to assist with the day to day usage of RAYTRACE. The greater theoretical detail that many researchers may wish to see is contained in a paper that is being published in Radio Science [1]. This user's guide is divided into two sections plus the appendices. The first section is introductory in nature and includes some background material, a 'packing list' of the files, and instructions for installation and setup. The second section is devoted to educating the user in the day to day operation of RAYTRACE. The appendices incorporate a variety of useful information.

The authors of the program RAYTRACE are Dr. Michael H. Reilly of the Naval Research Laboratory, and Dr. Eric L. Strobel of Interferometrics Inc. Dr. Reilly may be reached at the following address:

Dr. Michael H. Reilly
Code 4180.2
Naval Research Laboratory
Washington DC 20375

Phone: (202) 767-2891.

Dr. Strobel may be reached at:

Dr. Eric L. Strobel
Interferometrics Inc.
8150 Leesburg Pike
Vienna VA 22180

Phone: (703) 790-8500.

Summary of capabilities

The program RAYTRACE has a number of distinctive capabilities. It performs fully three-dimensional HF/VHF raytracing through a climatological ionosphere. (Magnetic field effects are, however, not currently implemented.) Single or multiple rays may be traced during a single run, allowing the user greater flexibility in modeling. The

size of the increments used in the raypath calculation is user adjustable so that appropriate compromises may be made between execution time and accuracy. The program can trace rays between points at different altitudes. Cutoff criteria may be set to stop the calculation at a particular range or altitude.

RAYTRACE calculates a number of important quantities about the ray. The change in signal intensity due to distance and ionospheric focusing or defocusing is calculated. The group and phase path length values are both calculated. The location, as well as information on the direction of the ray, are calculated for the end point of the ray and any intervening earth impact points (collectively these are called the data points for the ray).

Technical background

The RAYTRACE program is an implementation of a technique that is much more fully discussed in a paper that will be published in Radio Science [1]. A brief synopsis of that paper's discussion concerning program details is presented here. This section may be skipped with no loss of continuity.

The RAYTRACE algorithm utilizes a set of raypath differential equations that is separable under some approximation assumptions. These assumptions involve expanding the square of the index of refraction in a Taylor series in a local set of Cartesian coordinates and then truncating the series. If the coefficients of this expansion are known, then the raypath differential equations are integrable. The coefficients corresponding to the terms in the horizontal direction are obtained by spatial interpolation of sets of the coefficients defined on a latitude and longitude grid. Those in the vertical direction are obtained by using the functional form assumed by whatever particular ionospheric model the raytracing program has been customized to work with.

Given the truncation of the Taylor series, the equations for the raypath are only valid for a limited distance from the origin of the Taylor expansion. For this reason, the modeled raypath is built up incrementally. The raypath equations, then, give the coordinates of the end point of a ray increment in the local Cartesian system.

The program utilizes an ionospheric specification on a latitude-longitude grid of points. The ray is constructed by successively calculating the coefficients of the series, evaluating the raypath equations, and updating the coordinates and other needed values in preparation for the next increment. The process continues until some cutoff criterion is met.

This raytracing algorithm is quite compact, as it utilizes analytical integration of the raypath differential equations, rather than numerical integration. The storage requirements are smaller than if numerical integration was used because a particular (perhaps piecewise) functional form for the vertical structure of the ionosphere is assumed. This exacts a price, however. If a change is made to the underlying ionospheric model, or if a different model is to be used, portions of the RAYTRACE code need to be changed. The current implementation of RAYTRACE corresponds to that detailed in the paper [1], so the model to which it is currently matched is the RADAR-C ionospheric model [2]. The vertical profile from this model and the number of parameters necessary to specify the profile determine the calculation of the expansion coefficients and the resulting parameters and derivatives.

Operating requirements

The current implementation of RAYTRACE is written in Fortran 77. Subject to very minor compiler dependencies, which are discussed in later sections, the program will run on most computers having Fortran 77 compilers. The total amount of disk storage recommended for use with RAYTRACE is about one megabyte. Given the extra space needed for the compiler and system files, use of a hard disk is virtually essential for microcomputer users of RAYTRACE. If necessary, it is possible to place the executable version of the program, the associated files, and a data file on a floppy disk. A minimum of 512 K of memory is recommended, with 640 K or greater suggested. The presence of a math coprocessor is essentially a requirement for microcomputer users.

An additional requirement that many users will face is a method of generating input to RAYTRACE. This will entail a program that will produce a file, with a format given in Appendix A, from either an iono-

spheric model or experimental data. These data may be entered by hand, but this method of data input to RAYTRACE is only practical for relatively small amounts of data.

Design philosophy

The RAYTRACE program has been developed to meet specific user needs, however the development process has been geared toward maintaining the greatest degree of generality in operation consistent with the original specific needs. For this reason, the program is user driven to the greatest extent possible. This provides some capability to do calculations in a "What if ... ?" manner if desired. The current version of the program represents an attempt to strike a balance between modularity, maintainability of the code, and compactness of both memory space and user interface.

Part of the basic outlook taken while developing RAYTRACE is that this program will most certainly be put to unanticipated uses. The program is therefore set up to be as straightforward as possible (at least as simulation programs like this go) to modify for such uses. Some ideas on customization are given later. Such customization is bound to occur as RAYTRACE becomes more widely disseminated. The program will grow, but some questions will inevitably come back to the original authors of RAYTRACE. It is therefore strongly recommended that any variations, changes, or additions to RAYTRACE be commented as profusely as possible in the source code. Additionally, the program's authors (and, in fact, the research community at large) ought to be kept abreast of such alterations when appropriate. If this occurs, it will not only aid RAYTRACE's authors in responding to questions, but will facilitate the evolutionary improvement of the program as a general research tool.

Conventions used

It is assumed that the user is familiar with the operating system for the machine that is used. Additionally, the user ought to be knowledgeable about the operation of the compiler and editor to be used. Filenames are given in all capital letters (FILE.EXT), with a three letter extension. The total length of filenames is not to exceed ten characters. Prompts are generally displayed in double quotes (" "), with responses given in single quotes (' '). The quotes are not to be typed.

RAYTRACE is currently implemented on a DOS system and a VAX/VMS system, with subtle differences between the two implementations. This manual is written so that those with DOS systems see as little as possible about VMS systems, and vice versa. Although this approach results in some overall repetition, the user may skip over sections not dealing with his/her system without fear of missing key material.

FILES

What's included

The RAYTRACE package is best transported as source code which may then be copied onto a system meeting the requirements stated above for production of the executable version. What follows are brief descriptions of the RAYTRACE source code files and the various supporting files.

The source for the RAYTRACE program itself is broken into six files. The file RAYTRA4.FOR is essentially the user interface shell within which the actual raytracing routines are imbedded. The handling of access of data files is also within this set of routines. RRAYSB.FOR is the actual meat of the raytracing. This contains the routine which implements the raytracing algorithm. The ionospheric parameters and gradients necessary for the raytracing are calculated by routines in RIONO.FOR. Routines to support the calculation of phase path length information are contained in RPHASE.FOR. The boundaries between segments of the vertical profile of the ionosphere will, in general, be tilted with respect to local vertical. The routines to take this tilting into account are contained in RTILT.FOR. Various other supporting routines are located in the remaining source file, RMISC.FOR.

When RAYTRACE is run, it looks to a few text files which contain information on the make up of the menus that the user will see. These are the files with the .MEN suffix. MASTER.MEN, as the name would indicate, is the master file that RAYTRACE looks for to determine what other menu files will be needed. MAIN.MEN contains the information on the construction of the main menu that the user interacts with during a raytracing run. LAUNCH.MEN and ELEV.MEN determine the menus by which the user will input parameters used in the raytracing.

One more file is included. READER2.FOR is the source code for a brief program that may be used to display the results of a raytracing. Since analysis needs will vary from user to user, this program is meant to provide the basis for more sophisticated programs to analyze ray-tracing results.

Table 1

Table of files

Filename	Size (bytes)
RAYTRA4.FOR	50248
RIONO.FOR	63071
RMISC.FOR	42064
RPHASE.FOR	18811
RRAYSB.FOR	27181
RTILT.FOR	29584
ELEV.MEN	237
LAUNCH.MEN	237
MAIN.MEN	163
MASTER.MEN	47
READER2.FOR	3477

Formats available

RAYTRACE has been, to date, primarily used on a VAX under the VMS operating system, and on an AT-compatible microcomputer running MS-DOS. The program can be furnished in formats compatible with either of these two environments. It may also be possible to furnish it in other formats upon request.

PROGRAM SETUP

VMS environment

Setup and installation of RAYTRACE in the VMS environment will be made more convenient by appropriate planning of directories. What follows is one possible directory setup that has been of use while developing the program. Individual user needs and pre-existing directories will dictate the directory structuring for each site. The pri-

mary point is that the user of RAYTRACE will make usage easier by planning ahead.

The directory scheme used during development and testing of RAYTRACE involved the use of three parallel directories. A working directory was set up containing the source code. Editing, compiling, and linking are performed within this directory. When significant amounts of debugging are anticipated, the .MEN files and a trial input file are moved into the working directory, although by and large such duplication of files is avoided in order to save space.

A simulation directory is maintained for day-to-day usage of RAYTRACE. This directory holds .DAT (input) and the .MEN files, as well as the executable version of RAYTRACE. Numerous input files may be maintained by naming them in a mnemonic fashion.

The simulation directory is maintained in an uncluttered state by having a results directory to which the results of raytracings may be moved. This is particularly important because RAYTRACE simply uses generic filenames when it writes out results. If many runs are to be performed, it is to the user's advantage to rename and move the result files in order to keep them straight.

An additional directory that may be of help to some users is a versions directory. This may be a subdirectory of the working directory, to which older or different versions of the code are moved. Again, these files should be renamed in order to distinguish them at a later date. An advantage of such a subdirectory is keeping the working directory uncluttered.

The source code may now be copied into the desired directory. If the source is uploaded to the VMS system from a DOS system, some things may need to be edited before RAYTRACE will successfully compile. Some DOS-based Fortran compilers need markers at the beginning of subroutines so that subroutines will start on a new page when the compiler is asked to produce a listing file. These markers need to be removed.

Also, different compilers have different means of suppressing generation of a new line after a WRITE statement to the screen. This means that three lines of the subroutine KEYBRD need to be changed from one development environment to the next. This subroutine is located in the file RAYTRA4.FOR. Edit this file and go to the subroutine KEYBRD. There is a comment block below the variable declarations which describes the necessary changes.

RAYTRACE is now ready to be compiled and linked. If default naming is to be used, then RAYTRA4 should be listed first in the link command. The executable version may now be copied to the raytracing directory. The .OBJ files should be left intact. This way, when changes are made to one source file, only that one file need be recompiled and all the object files may be linked. If disk space is a consideration, the executable may certainly be deleted from the working directory.

READER2.FOR should also be compiled and linked. The executable may be copied to the results directory. The object and executable files left in the working directory may be deleted to conserve disk space.

The following files should now exist. The working directory should contain a complete set of both source and object files for RAYTRACE, as well as the source for READER2.FOR. The executable versions of RAYTRACE (default name: RAYTRA4.EXE) and READER2 should be in the locations that the user has chosen as appropriate. The .MEN files should be located in the same directory as the RAYTRACE executable.

The process of compiling, linking, and moving these files will be made considerably easier by making use of the capability of VMS to use .COM files. This is especially true if modifications to the programs are intended.

DOS environment

Setup and installation of RAYTRACE in the DOS environment will be made more convenient by appropriate planning of directories. What follows is one possible directory setup that has been of use while developing the program. Individual user needs and pre-existing directories will dictate the directory structuring for each site. The primary point is that the user of RAYTRACE will make usage easier by planning ahead.

The directory scheme used during development and testing of RAYTRACE involved the use of three parallel directories. A working directory was set up containing the source code. Editing, compiling, and linking are performed within this directory. When significant amounts of debugging are anticipated, the .MEN files and a trial input

file are moved into the working directory, although by and large such duplication of files is avoided in order to save space.

A simulation directory is maintained for day-to-day usage of RAYTRACE. This directory holds .DAT (input) and the .MEN files, as well as the executable version of RAYTRACE. Numerous input files may be maintained by naming them in some mnemonic fashion. This may become crucial, as with a large number of files it is entirely possible to accidentally destroy another file by attempting to have two files of the same name.

The simulation directory is maintained in an uncluttered state by having a results directory to which the results of raytracings may be moved. This is particularly important because RAYTRACE simply uses generic filenames when it writes out results. If many runs are to be performed, it is to the user's advantage to rename and move the result files in order to keep them straight. Again, this helps avoid the unpleasantness of unwanted destruction of result files.

An additional directory that may be of help to some users is a versions directory. This may be a subdirectory of the working directory, to which older or different versions of the code are moved. Again, these files should be renamed in order to distinguish them at a later date. An advantage of such a subdirectory is keeping the working directory uncluttered.

The source code may now be copied into the desired directory. If the source is downloaded to the DOS system from a VMS system, some things may need to be edited before RAYTRACE will successfully compile. Some DOS-based Fortran compilers need markers at the beginning of subroutines so that subroutines will start on a new page when the compiler is asked to produce a listing file. These markers need to be added.

Also, different compilers have different means of suppressing generation of a new line after a WRITE statement to the screen. This means that three lines of the subroutine KEYBRD need to be changed from one development environment to the next. This subroutine is located in the file RAYTRA4.FOR. Edit this file and go to the subroutine KEYBRD. There is a comment block below the variable declarations which describes the necessary changes.

RAYTRACE is now ready to be compiled and linked. If default naming is to be used, then RAYTRA4 should be listed first in the link com-

mand. The executable version may now be copied to the raytracing directory. The .OBJ files should be left intact. This way, when changes are made to one source file, only that one file need be recompiled and all the object files may be linked. If disk space is a consideration, the executable may certainly be deleted from the working directory.

READER2.FOR should also be compiled and linked. The executable may be copied to the results directory. The object and executable files left in the working directory may be deleted to conserve disk space.

The following files should now exist. The working directory should contain a complete set of both source and object files for RAYTRACE, as well as the source for READER2.FOR. The executable versions of RAYTRACE (default name RAYTRA4.EXE) and READER2 should be in the locations that the user has chosen as appropriate. The .MEN files should be located in the same directory as the RAYTRACE executable.

There are several utilities available with some DOS-based compilers that ought to be taken advantage of if available. One of the drawbacks of most DOS Fortrans is the colossal size to which the executable file grows, relative to the size of the source or object files. A necessary utility is therefore some sort of program that compresses the executable to the size that it ought to be.

The other utility is a 'make' utility. This automates the process of building that executable by checking the files that go into constructing the executable to see which ones have changed. Compilation only occurs for those files that need it. If much alteration of RAYTRACE is to be done, a 'make' utility will save a great deal of time. If a 'make' utility is not available, then creative use of DOS batch files may serve as a substitute.

PART TWO:

OPERATOR'S GUIDE

GENERAL INSTRUCTIONS

Overall program flow

The general scheme for using RAYTRACE is much like that of other simulation programs. Input is usually from a data file, and the results are stored in output files. During the course of execution, the user is given the opportunity to change a number of parameters, allowing a variety of situations to be simulated in one session. These settings may be saved for later use. The user is also given the ability to redo entries which were made in error.

Figures 1a & 1b summarize the overall flow of RAYTRACE. When the program is started, the user may choose between doing a completely new problem, or one which has been done before and therefore has an already existing data file with options set. If an old problem is to be worked on, all of the problem's option settings and ionospheric data are loaded in and the user is deposited directly into the main menu. If a new problem is being performed, the ionospheric data must be input. This will usually be by means of a grid file. A grid file is simply a file containing the pertinent ionospheric information at a gridded set of latitudes and longitudes. Grids may be entered from the console, but this is only practical for relatively small grids. Again, after the ionospheric data is entered, the user is placed in the main menu.

FIGURE 1a.

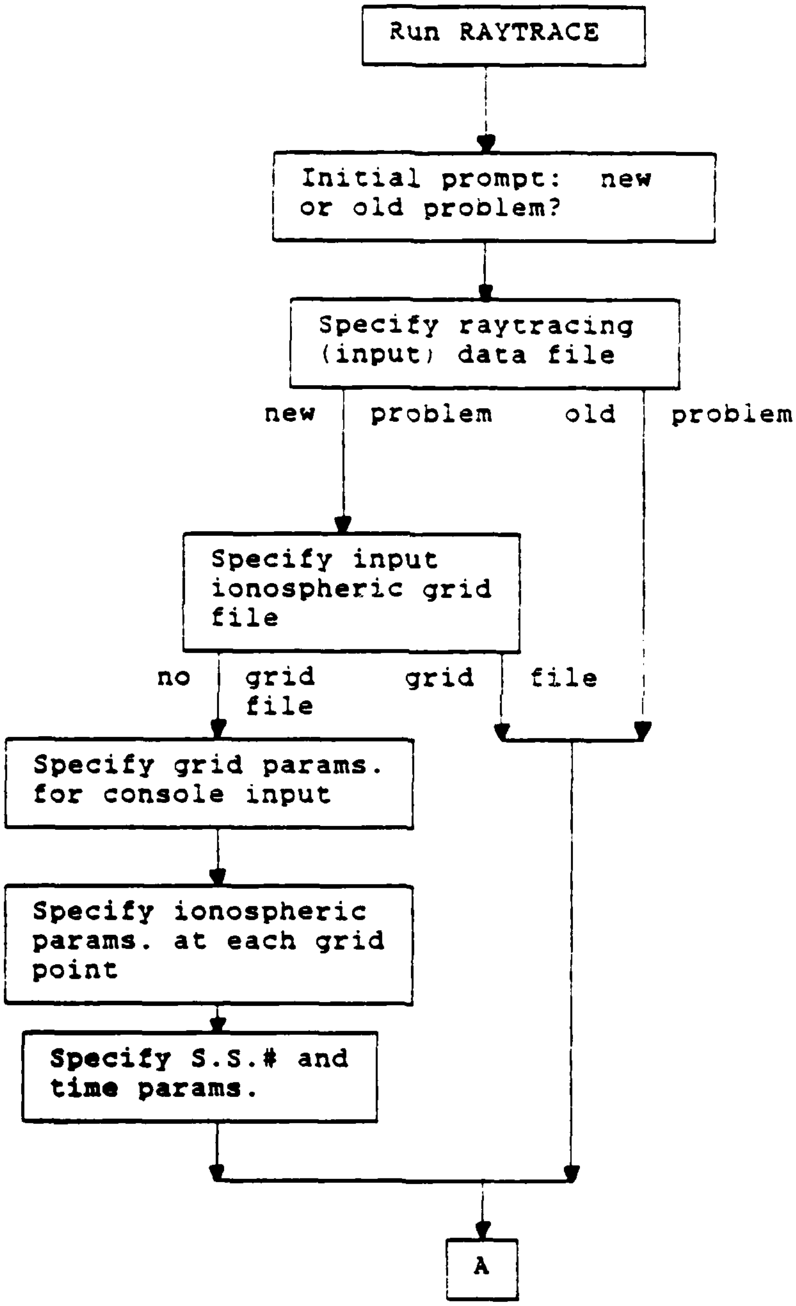
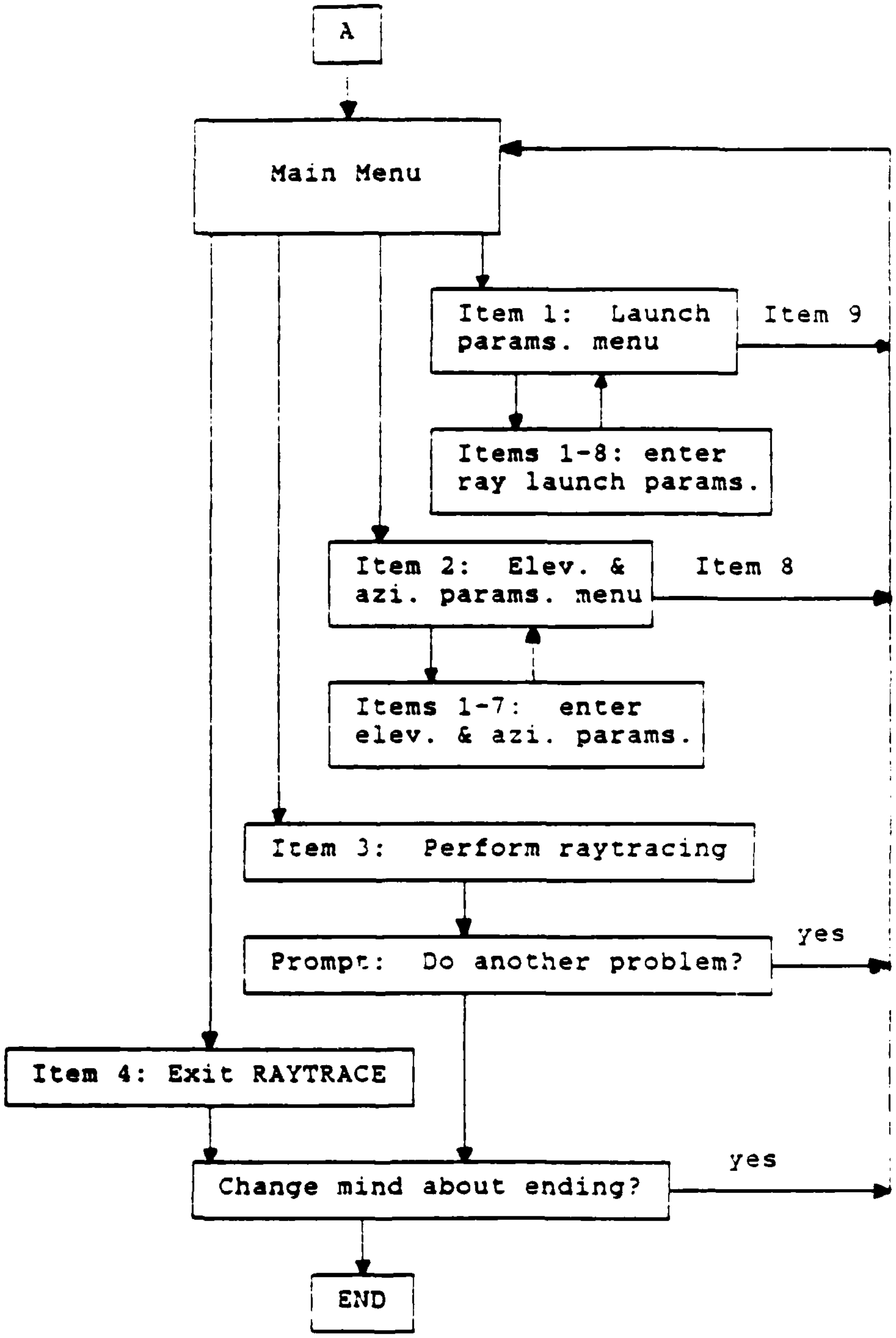

Run RAYTRACE
Initial prompt: new or old problem?
Specify raytracing (input) data file
new problem
old problem
Specify input ionospheric grid file
no grid file
grid file
Specify grid params. for console input
Specify ionospheric params. at each grid point
Specify S.S.# and time params.
A

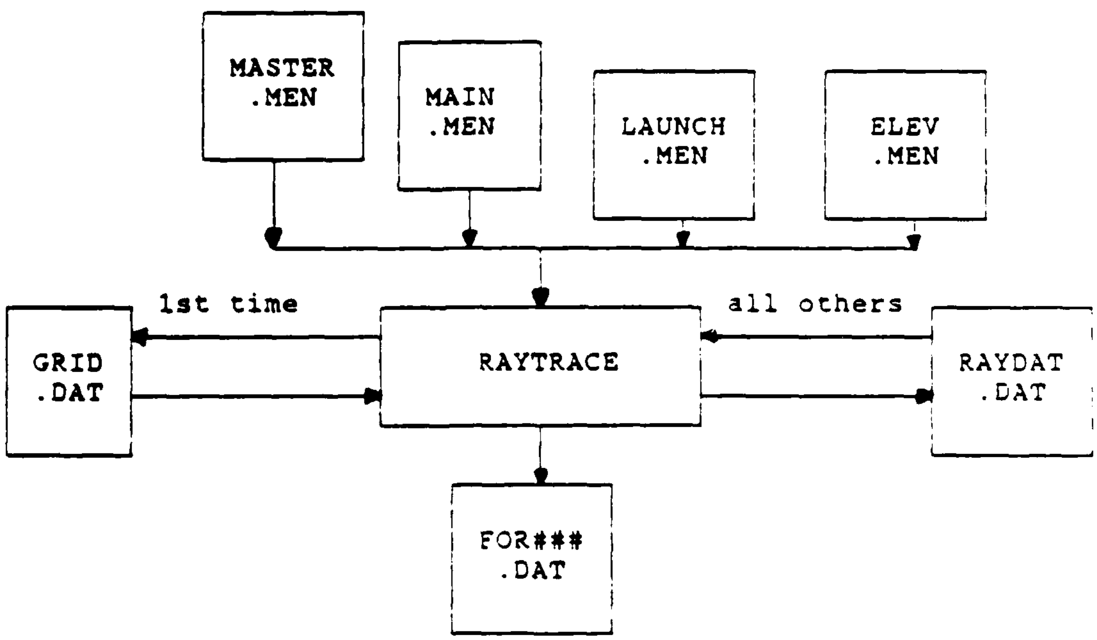
FIGURE 1b.


A
Main Menu
Item 1: Launch params. menu
Item 9
Items 1-8: enter ray launch params.
Item 2: Elev. & azi. params. menu
Item 8
Items 1-7: enter elev. & azi. params.
Item 3: Perform raytracing
Prompt: Do another problem?
yes
Item 4: Exit RAYTRACE
Change mind about ending?
yes
END

The main menu of the program is a central point to which execution will return unless the session is explicitly ended. There are two menu items which lead to other menus. These are for the entry of parameter settings. The other selections are to execute the raytracing using the current parameter settings, and to exit from the program. All of this will be discussed in greater detail below.

During execution, RAYTRACE must deal with a number of other files. The general relationship of the program and these files is summarized in Figure 2. The file GRID.DAT is a grid file, as described briefly above. It contains the ionospheric specification for a given situation and is used as input on fresh problems. This file must be generated by some outside means, although if the specification is entered into RAYTRACE by hand, a GRID.DAT file will be produced. The RAYDAT.DAT file is the file which contains the settings from the previous sessions, as well as the ionospheric specification. The four files with the .MEN suffixes determine the appearance and text of the menus. Results are output into files of the type FOR###.DAT, where ### starts at 040 (This choice is an artifact of earlier versions.) and increments each time a new raytracing is done within a given session.

FIGURE 2.



Ionospheric input

One of the primary benefits of RAYTRACE is the ability to do realistic three-dimensional raytracing with a complicated ionospheric specification. This capability is implemented by means of the grid file(s) described briefly above. The ionospheric grid file may be produced by a numerical model, or the appropriate parameters may be obtained from experimental data and massaged into a grid file. A grid file may also be produced by direct keyboard entry of ionospheric parameters into RAYTRACE.

The grid file, however it is produced, is composed of a number of data arrays. First, the definition of the latitude and longitude grid is recorded. Following this, the six ionospheric parameters required for specification of the vertical profile are given for each point on the grid. Finally, other pertinent data are recorded. The six parameters are the maximum plasma frequencies for the E, F1, and F2 layers, the heights for the maximum frequencies for the F1 and F2 layers, and the semithickness of the F2 layer. The choice of these parameters is determined by the profile model that is built into the current version of RAYTRACE. Currently, RAYTRACE uses the vertical ionospheric profile contained in the RADAR-C model [2]. Because the raytracing is so intimately mated with this model, any input data coming from experiment or some other model must be converted into this form or RAYTRACE must be modified to accommodate the new model profile. The full details on the composition of a grid file are found in Appendix A.

Operating instructions

Before beginning operation of RAYTRACE, be sure that the instructions in the setup section above have been followed. Also, be sure that any input data that are desired exist in a grid file. RAYTRACE may now be executed in the manner appropriate to the system it is operating on.

When RAYTRACE is first run, the following prompt appears:

" Is this a new (1) or old (0) problem? (0): ".

This prompt is typical of those used in RAYTRACE and so it is appropriate to discuss it here in some detail. First, the prompt displays some text which describes the information sought. Any non-intuitive form of response, such as the 0 or 1 above, or other information which will in some way limit the acceptable response, is displayed as part of the prompt text. After the prompt text, the current (or default) value is displayed in parenthesis. The colon marks the end of the prompt, after which the user types his response. If the user merely types a return, then the default value is used as the response. REAL-valued responses may be typed without the decimal point if there is no fractional part to the response, and character responses are case-insensitive.

The above prompt is used to determine whether or not this RAYTRACE run is for an entirely new data set, or a previously used data set. A new problem is simply one for which no RAYDAT-type file exists, or for which no previously existing files of this type will be used. This type of file has been mentioned briefly already and will be discussed more fully below. By selecting the option for a new problem the user is therefore able to start either a genuinely new problem, or use an old ionospheric specification with all new options. An old problem, on the other hand, is one for which the ionospheric specification and the options have been stored in a file of the RAYDAT type. Please note that a response to this prompt MUST be TYPED, i.e. the default cannot be accepted. This is an attempt to ensure that the operator will make the correct choice.

The next prompt that comes up is the following:

```
" Enter filename for storage of ionospheric
  information (10 char max):                (RAYDAT.DAT): ".
```

This prompt asks for the name of the file which will contain both the ionospheric specification and the parameter options settings used in the particular problem. If the problem is a new one and the file given already exists, a Fortran error message will occur, the content of which will depend upon the compiler used. Aside from this, the response to this prompt is straightforward.

The program will now proceed to the main menu if the problem has been declared to be an old one. However, the ionospheric specification still needs to be obtained if this is a new problem. The next prompt asks for the name of the file containing this ionospheric information:

```
" The ionospheric grid file name is?
  (type NONE if none exists)                    (GRID.DAT   ): ".
```

The data is retrieved from the grid file whose name is given and the program then proceeds to the main menu. If 'none' is typed (remember, the response is case-insensitive), the program then executes a section of code to read the appropriate information from the console.

Console entry of the ionospheric specification involves three sets of prompts. The first set of prompts asks the user to provide the necessary parameters for specifying the locations of the grid points. The prompts are:

```
"Input lat grid spacing (deg):              (      .000000000     ):
 Input lon grid spacing (deg):              (      .000000000     ):
 Input starting latitude and longitude in degrees.
                  LATITUDE :                (      .000000000     ):
 LONGITUDE (east = positive):               (      .000000000     ):
 Input # of grid points in lat.:            (      .000000000     ):
                     ... in lon.:           (      .000000000     ):
 Grid setup OK (Y/N) ?                      (         Y): ".
```

The first two queries set the latitude and longitude spacing between grid locations. The next two fix the location of the south-west corner of the grid. Longitudes are measured from Greenwich with positive values eastward and negative values westward, so that the range is between -180 and +180 degrees. The last pair of queries gives the extent of the grid in increments of the grid spacing in the appropriate direction. One word of warning is appropriate here. Don't forget about counting BOTH end points when entering the number of grid points! Finally, the user is asked if the values entered are correct. If so, the execution of the program goes on. If not, the user is returned to the beginning of this set of questions and allowed to repeat the data entry. The current settings will be shown as the default values which eases the correction of the errant entry.

The next set of prompts is used to obtain the actual vertical profile data at each grid location. The location is displayed and the user is prompted to enter each of the six values needed to specify the vertical profile used with this version of RAYTRACE, as follows:

```
" GRID PT. - Lat:      ##.####        - Lon:      ###.####

Input  foE**2 :                        (       .000000000     ):
Input     hmF1 :                       (       .000000000     ):
Input foF1**2 :                        (       .000000000     ):
Input     YmF2 :                       (       .000000000     ):
Input     hmF2 :                       (       .000000000     ):
Input foF2**2 :                        (       .000000000     ):
Profile inputs OK (Y/N)?               (          Y):  ".
```

The values requested are the plasma frequencies (squared) for the three ionospheric layers, the E, F1, and F2, at the layer maximum. These are given in units of MHz-squared. The heights of the maximum plasma frequency for the F1 and F2 layers are given in kilometers as is the value of the semithickness of the F2 layer. As with the previous set of prompts, if the user acknowledges that the inputs are OK, the program will continue on. If the values entered are not correct, the user is given the chance to reenter them. These profile entry prompts continue to come up on the console until all grid locations have been exhausted.

The final of the three sets of prompts asks for the time of the simulation and the sunspot number:

```
" Input (integer) sunspot number:        (    0):
  Input year:                            (    0):
  ...  month:                            (    0):
  ...    day:                            (    0):
  Input UT time (hr) :                   (    0):
  ...   UT time (min):                   (    0):
  S.S. # and times OK (Y/N)?             (          Y):  ".
```

The sunspot number is the Zurich sunspot number. The year, month, and day are given as their usual integer values. The time is Universal Time and is integer also. After this prompt, the entry of the ionospheric specification is complete and the program proceeds to the main menu.

By whatever route, the program will now have reached the main menu. The main menu serves as the hub of the program as far as the user is concerned. All major actions that may be performed are done from the main menu. The menu contains four items:

```
                              MAIN MENU

          1 - Edit launch parameters

          2 - Edit elevation/azimuth parameters

          3 - Proceed with raytracing

          4 - Quit program

      Your choice (1-4 only, please)?           (    3):
       Press RETURN to continue.                (          ): ".
```

This exhibits the typical structure of a menu in RAYTRACE. The menu is titled, and the items are given with corresponding numbers. Selection is made by entry of an item's number. The entry is checked and if it is out of bounds for that menu, the user is prompted for the choice again. The 'Press RETURN to continue.' prompt is self-explanatory, except for the opportunity it affords the user to abort the previous action. If the user types 'a' (for abort), the menu choice will be aborted and the menu will be presented again.

The first two items of the main menu are used to invoke menus for the entry of parameter values for the particular problem being done. These values are divided up into basically general values in the first menu and elevation/azimuth values in the second. The third and fourth items are self-explanatory.

Selection of item 1 of the main menu brings the Launch Parameters menu to the screen. The parameters which are entered by means of this menu are of a general sort. They include the starting location of the ray, the properties of the ray, and the conditions for cutting off the ray calculation. The Launch Parameters menu is:

"   LAUNCH PARAMETERS MENU

```
1 - Bounce limit

2 - Signal intensity

3 - Conductivity

4 - Launch point

5 - Launch height

6 - Range & height limits

7 - Ray path increment

8 - Wave frequency

9 - Done

Your choice (1-9 only, please)?          (    0):
 Press RETURN to continue.               (            ): ".
```

The first item on the menu is the bounce limit. Selection of this item brings up the following prompt:

```
" Bounce limit                              (    1): ".
```

When the ray propagation represents traditional HF communications type raypaths, i.e. rays which are reflected from the bottom side of the ionosphere and which return to earth, it is convenient to place a limit on the number of times that a ray can return to earth. This may be a maximum of 10 currently. In essence, this provides for a kind of fuzzy range cutoff to supplement the hard range cutoff which will be discussed later. Pertinent values for the ray are output to a file at bounce (earth impact) points, and at the end of the ray. This item is somewhat meaningless for raypaths which are not restricted to the space between the bottom of the ionosphere and the earth's surface. One artifact of this is the blank record which may appear at the end of the results file for certain raypaths. This blank record is completely harmless and, to use current terminology, it's not a bug, it's a feature.

The next two items deal with the signal intensity calculations that will be done along the ray. The prompt for item 2:

```
" Do signal intensity (no = 1)?             (    0): ",
```

is used to enter a flag which turns the calculation of signal intensity on and off. Currently, the signal intensity computation that is performed is the geometric spreading of a ray bundle, including focusing and defocusing effects, and reflection loss upon earth impact. The third prompt:

```
" Reflect from ground(1) or water(0)?      (    0): ",
```

is used to determine the conductivity of the reflection surface used in the calculation of reflection loss. At present this is a global choice.

Menu selections 4 and 5 determine the geographic location and altitude of the starting point of the ray. The latitude and longitude are entered at the prompt:

```
" Enter launch pt. latitude              (      .000000000    ):
  Enter launch pt. longitude             (      .000000000    ):".
```

The values are entered in degrees with the longitude ranging from -180 to +180 degrees, positive values to the east of the prime meridian. The starting height is entered at the prompt:

```
" Enter starting ht. (km)                (      .000000000    ):".
```

This is the height of the starting point above the surface of the earth, defined as the spherical surface having a radius equal to the average earth radius.

Item six provides for the entry of cutoff criteria for the ray-tracing problem:

```
" Enter range limit (km)                 (      3000.00000    ):
  Enter altitude limit (km)              (      36000.0000    ): ".
```

The range in the range limit prompt is great circle range at the earth's surface. It should be thought of as an angular range along the great circle direction defined by the original azimuth of the ray, since this is how it is handled internally. By thinking of this limit as an angular one, it may be readily generalized to problems for which both end points of the ray are above the earth's surface. The altitude limit simply sets a cutoff altitude above the surface of the earth. The default value of 36000 km represents the altitude of geosynchronous satellites.

Selection seven of the Launch Parameters menu allows the user to set the size of the steps taken along the ray as its propagation is modeled. The prompt is as follows:

" Enter raypath increment (km) (4.00000000): ".

Settings for this represent a compromise between accuracy of the results and speed of computation. The default value of 4 km represents a rough optimum. Practically, values much smaller than 0.1 km cause the computation to take much too long and sufficiently small values can begin to cause a decrease in accuracy due to accumulation of roundoff errors from all over the program. Large values are limited by the theory underlying RAYTRACE, and should generally be less than 10-15 km.

The eighth menu item sets the frequency of the electromagnetic wave that the ray represents. The prompt is:

" Enter wave frequency (MHz) (5.00000000): ".

Because effects due to the earth's magnetic field are not implemented in the current version of RAYTRACE, use of frequencies much below about 2 or 3 MHz is questionable. At the high end, frequencies beyond a few GHz begin to cause inaccuracies because of repeated operations involving one minus a small number.

Selection of item nine in the Launch Parameters menu returns the user to the main menu. If the user has decided to alter any of the parameters set in the Launch Parameters menu, the menu may simply be reentered at this point and the values changed. Parameter settings for the raytracing may be changed over and over, the ray will be traced using the current settings.

The second item in the main menu invokes the Elevation Parameters menu. This menu has eight items:

"   ELEVATION PARAMETERS MENU

```
        1 - Dimension of problem

        2 - Starting azimuth

        3 - Starting elevation

        4 - Elevation resolution

        5 - Azimuth resolution

        6 - Elevation limit

        7 - Azimuth limit

        8 - Done

    Your choice (1-8 only, please)?          (    0):
     Press RETURN to continue.               (           ): ".
```

This menu works in an identical fashion to the Launch Parameters menu, but with one twist. It will become clear from discussion of the individual menu items that some of them simply do not apply in all circumstances. In cases where this is true, those items which don't apply are disabled. Attempting to select such a disabled item will return the user to the menu.

Selection one of the Elevation Parameters menu allows entry of a quantity that is known as the dimension of the problem:

```
" Enter dimension of problem          (       .000000000    ): ".
```

The dimension may have values of 0, 1, or 2. The dimensionality is that of the pattern of rays to be sent out by RAYTRACE. A single ray may be sent (dimension 0). However, multiple rays may be sent in a single run. A vertical fan of rays may be specified, which is to say that the program may be instructed to step in elevation for a fixed azimuth. This is dimension 1. The program may be instructed to step both in elevation and azimuth. This is dimension 2. The aimpoints of the ray(s), if pictured on an elevation versus azimuth plot, then form either a one or two dimensional array, or a single point.

Continuing with this visualization, the array is taken to start in the lower left corner, that is, the lowest value of both elevation and azimuth. For a single ray, the array of values is simply restricted to just this single point and the dimension one problem uses the leftmost

column of the array. All cases require the input values for the starting corner of the array of elevation and azimuth values. Menu items two and three handle this:

" Enter starting azimuth (deg) (.000000000): ",

and,

" Enter starting elevation (deg), use neg.
for values >90 deg from zenith. (.000000000): ".

The azimuth is given in the conventional manner. As the prompt for starting elevation states, elevation angles below horizontal are allowed. Acceptable values range from -90 degrees to +90 degrees, permitting rays which have their origin at altitude to be sent downward.

The next menu item, item 4, is used to enter the elevation spacing between rays,

" Enter elev. resolution (deg) (.000000000): ".

This only applies when a multi-dimensional array of rays is being considered, and so may only be selected when the problem dimension has been declared to be one or two. The elevation values are stepped upward (i.e. toward increasing elevation) by this value. Related to this is item six, where the limiting value for the elevation is given. This prompt,

" Enter elev. limit (deg) (.000000000): ",

is used to define the extent of the pattern of rays in elevation. As with the previous prompt, this may only be selected when the problem is declared to be of dimension one or two.

The fifth prompt is used for entry of the azimuthal spacing in the pattern of rays to be sent out. The prompt is:

" Enter azimuth resolution (deg) (.000000000): ".

This prompt is coupled with the seventh prompt,

" Enter azimuthal limit (deg) (.000000000): "

which provides the program with the information on the azimuthal extent of the ray pattern. These selections only apply for the case when the dimension of the problem is two.

Finally, the eighth item of the Elevation Parameters menu will return the user to the main menu when selected. As with the Launch

Parameters menu, the user may continue to change the values entered in the Elevation Parameters menu until they are satisfactory. These are then the values which will be used in the raytracing.

The main menu's third selection will cause the actual computation of the ray(s) to begin. The values entered in the Launch Parameters and the Elevation Parameters data entry menus are the values that will be used. At this point, if this is the first time through a new problem, the parameters and ionospheric data are written out to the RAYDAT-type file and the ray computation begins immediately. The first time through an old problem, the user is prompted:

" Overwrite existing file (Y/N)? ( ): ",

about overwriting the old parameter values with the new ones. If not, the user is asked:

" Backup the datafile (Y/N)? ( ): ",

whether a backup copy (another RAYDAT-type file) of the settings is desired. After answering these questions, the ray calculation begins. For either case, after the initial time in a problem, every succeeding time the user is asked:

" Update the datafile (Y/N)? ( ): ",

whether or not the data file should be updated with the latest settings. The interface is set up to give the user ample opportunity to preserve the new parameter settings if desired, but the user has the choice of not saving these settings if that is what is desired.

The user is informed of the start and finish of the ray calculations by the messages:

" BEGINNING RAY LOOPS.

RAY LOOPS DONE. ".

Upon completion of the computations, the user is informed of the file that the results have been written to by the following message:

" Results written to file FOR040.DAT ".

The numeric part of the filename will increment each time the user does another raytracing in a single session. Leaving the program resets the numeric portion of the filename.

Now that a particular raytracing is done and the results are stored away, the user is prompted:

```
" Do another problem (Y/N)?              (          ): ",
```

whether or not additional raytracing is desired. An affirmative answer will return the user to the main menu so that any changes to parameters may be made and more rays may be traced. A negative answer has the same effect as choosing item four from the main menu. First, the prompt:

```
" Press RETURN to continue.              (          ): ",
```

is issued, allowing the user to abort the exiting of the problem by entering 'a'. If a return is entered, then the program exits with the following final message

```
" Session done.  For safety's sake, copy the data
  and result files into a separate directory to
  prevent accidental overwriting.                  ".
```

This warning should be heeded. For use on VAX/VMS systems, a large number of virtually indistinguishable files with the same names and different version numbers may result in loss of results through confusion over which version is which. DOS based systems present an even greater problem, because the attempt to save results to a file already in existence will cause the old file to be destroyed.

Output Files

RAYTRACE produces three types of output files. The first is a GRID.DAT type of file, if the ionospheric specification has been entered by hand. This file has been discussed already. The second type of file is the RAYDAT.DAT type. This file is essentially just a GRID.DAT file with the additional user-entered parameters appended to it. First, the definition of the latitude and longitude grid is recorded. Following this, the six ionospheric parameters required for specification of the vertical profile are given for each point on the grid. Other pertinent data are recorded, including the date and time of the ionospheric specification, and the sunspot number. These previous data are just those recorded in the GRID.DAT file. Addition-

ally, the values from the Launch Parameters menu are recorded and finally the values from the Elevation Parameters menu are stored. Full detail of the RAYDAT.DAT type of file is given in Appendix A.

The remaining type of data file is the results file, FOR###.DAT. This file contains the results of a single run of RAYTRACE. First, the bounce limit (plus one for the end point of the ray), the number of azimuths, and the number of elevations are recorded. This enables any program reading the results to break the results out into those for individual rays. For each ray, the azimuth and elevation are recorded. Then the various results are saved for each data point along the ray. These data points are earth impact points and the end point. If the ray is terminated before the bounce limit is reached, blank records will occur and should be ignored. Each succeeding ray is recorded in a similar manner; first, azimuth and elevation, and then the results. The details of this particular file type are presented in Appendix A.

It should be restated that the user ought to be careful and develop defensive file maintenance procedures. After each session files should at the least be given more suitable names that are perhaps indicative of their contents. This will not only reduce the likelihood of accidental destruction of results, but will aid in later analysis. Also, the user should be aware of when the files get written to disk. The RAYTRACE produced GRID.DAT file has its contents written during the entry of the data. The RAYDAT.DAT file is written just before the rays are traced, and the FOR###.DAT (results) file is written during the raytracing.

TUTORIAL EXAMPLES

Although the operation of the various commands of RAYTRACE has now been discussed, this knowledge will be reinforced by going step by step through some examples. These examples will be presented with actual samples of screens, for clarity. The first tutorial involves setting up RAYTRACE to use a spherically symmetric ionosphere with the specification entered by the console. This will create the data files used in the subsequent two tutorials. The second is to perform a new problem using a grid file as input. The last is to perform an old problem. It is hoped that the user will actually perform these tutorials on his/her system, even if operation of the program seems obvious by this time. At the least, the exercise will increase familiarity with the operation of the program, while potential problems or questions may be preempted by use of these tutorials.

New session using keyboard input

To begin this tutorial, run RAYTRACE. This will be a new problem, so respond with '1' to the first prompt. The next prompt will ask about the file to which all the ionospheric and parameter information will be written. The filename that has been chosen for this tutorial is TRIAL.DAT. At this time the screen should look like the following.

Screen 1

```
Is this a new (1) or old (0) problem?   (   0):1

Enter filename for storage of ionospheric
information (10 char max):               (RAYDAT.DAT):trial.dat
```

The next prompt is the prompt which asks about the file from which the ionospheric specification will be read. For this example, there is none, so the appropriate response is 'none'. The screen that corresponds to this is Screen 2.

Screen 2

```
The ionospheric grid file name is?
 (type NONE if none exists)            (GRID.DAT  ):none
```

The next set of prompts that comes up is a set of prompts asking about the definition of the grid on which the ionosphere is to be specified. The first two prompts deal with the latitude and longitude spacing between the grid points. Since this example is a spherically symmetric case, only one grid point will be needed and so the spacing is immaterial. For this reason, a response of '1' is chosen in each case. The next two prompts ask for the starting latitude and longitude for the grid, in this case this is the location of the only grid point. The responses chosen are '45' for latitude and '-90' for longitude. This longitude corresponds to a west longitude. The final two informational prompts here ask about the size of the grid in the latitude and longitude directions. Since there is to be only one point, the number of points in each direction will be '1'. Finally, there is a prompt asking whether the input is correct. Assuming everything has been entered as shown, then typing return will accept the default choice of 'Y'. At this point, the screen should look like Screen 3.

Screen 3

```
Input lat grid spacing (deg):            (      .000000000    ):1
Input lon grid spacing (deg):            (      .000000000    ):1
Input starting latitude and longitude in degrees.
                  LATITUDE :             (      .000000000    ):45
LONGITUDE (east = positive):             (      .000000000    ):-90
Input # of grid points in lat.:          (      .000000000    ):1
                  ...  in lon.:          (      .000000000    ):1
Grid setup OK (Y/N) ?                    (         Y):
```

Now the program begins to prompt for the ionospheric information for the location specificied above. The location of the point is given at the top of the prompt. The parameters prompted for occur ordered in terms of height. It should be noted that the values given here are merely fictitious. First is the plasma frequency squared for the maximum of the E-layer, with a value to be entered as '2'. Next the F1-layer is specified, by the height of the maximum of the F1, '150', and the plasma frequency squared of the F1 maximum, '7'. Finally, three F2-layer parameters are prompted for. They are the semithickness of the F2, '65', the height of the F2 maximum, '300', and the plasma frequency squared at the maximum of the F2, '20'. Again, the program prompts to see if all the input is OK. If it is, the response should be the same as for the similar prompt above. The console screen should now look like Screen 4.

Screen 4

```
GRID PT. - Lat:      45.0000        - Lon:     -90.0000

Input  foE**2 :                           (       .000000000     ):2
Input    hmF1 :                           (       .000000000     ):150
Input foF1**2 :                           (       .000000000     ):7
Input    YmF2 :                           (       .000000000     ):65
Input    hmF2 :                           (       .000000000     ):300
Input foF2**2 :                           (       .000000000     ):20
Profile inputs OK (Y/N)?                  (          Y):
```

The final set of prompts for the environmental specification involves recording the date and time of the problem, and the sunspot number. First the sunspot number is entered, in this example '80'. Next, in descending order of time scale, come the date and time prompts. As indicated by the defaults given, these are expected to be integers. This example uses the following responses in order: '1988', '2', '17', '12', and '00'. After the verification prompt, the screen should look like Screen 5.

Screen 5

```
Input (integer) sunspot number:              (    0):80
Input year:                                  (    0):1988
...  month:                                  (    0):2
...    day:                                  (    0):17
Input UT time (hr) :                         (    0):12
...   UT time (min):                         (    0):00
S.S. # and times OK (Y/N)?                   (         Y):
```

At this point, the program brings the user to the main menu. It is from this menu that other menus are invoked for entry of the remaining parameters, and that the actions of tracing rays and quitting are taken. The parameters are currently a blank slate, so both of the data entry menus need to be selected and the parameters entered. Starting at the top, select menu '1'. The program will respond with another prompt. As it says, if the selection is the desired one, the user needs only type a return to proceed. As was mentioned earlier, though, the selection may be aborted by responding with an 'a'. If this is done, the user will be returned to the previous menu. Before typing return, the screen will look like Screen 6.

Screen 6

```
                    MAIN MENU

     1 - Edit launch parameters

     2 - Edit elevation/azimuth parameters

     3 - Proceed with raytracing

     4 - Quit program

Your choice (1-4 only, please)?          (    0):1
 Press RETURN to continue.               (          ):
```

Selection of item one of the main menu brings the user to the Launch Parameters menu. This menu is used for the entry of various parameters relating to the ray calculation. These values include the location of the starting point for the ray(s), settings which are used to halt the calculation, properties of the ray(s), and values which determine whether some calculations are performed and if so, how. This example will simply make the selections from the menu in numerical order, therefore, the first entry will be '1'. This situation is reflected in Screen 7.

Screen 7

LAUNCH PARAMETERS MENU

```
    1 - Bounce limit

    2 - Signal intensity

    3 - Conductivity

    4 - Launch point

    5 - Launch height

    6 - Range & height limits

    7 - Ray path increment

    8 - Wave frequency

    9 - Done

Your choice (1-9 only, please)?          (    0):1
 Press RETURN to continue.               (          ):
```

This selection from the Launch Parameters menu brings up a query about the limit to be placed on how many times a ray may return to earth. This is useful when dealing with low frequencies which will not penetrate the ionosphere. The problem then may be set up to terminate after a certain number of hops has occurred, regardless of the other termination conditions. For the purpose of this example, accepting the default value of '1' will be sufficient, so the response will be to just type a return. The screen should look like Screen 8. Note that in some of the screens that follow, only the lower portion of the screen is shown.

Screen 8

```
Bounce limit                                ( 1):
```

Once the entry of the data is made, the user is immediately returned to the Launch Parameters menu. This time the selection will be item '2'. Because the process by which items are selected from menus is uniform, the screen need not be repeated. Menu item two leads to a prompt which allows the program to know whether or not to perform signal intensity calculations. For the sake of speed in running the example, the value entered is '1' which corresponds to no. The screen example is given as Screen 9.

Screen 9

```
Do signal intensity (no = 1)?           ( 0):1
```

Returning to the Launch Parameters menu, the next selection made will be '3'. This item allows the user to designate a global value for the conductivity of the earth's surface. These values correspond to soil and water. They are average values. Conductivity values are used when signal intensities are calculated so that the loss due to reflection from the earth's surface may be accounted for. Since no signal intensity will be calculated for this example, this item was selected for instructional purposes only. Because any answer will suffice, the default may be accepted by typing return. This is shown in Screen 10.

Screen 10

```
Reflect from ground(1) or water(0)?     (    0):
```

The next selection from the Launch Parameters menu will be item number '4'. This prompts the user for the latitude and longitude of the launch site of the ray(s). The values are given in degrees. Remember that longitudes are to be given as positive to the east of Greenwich, with the prime meridian being 0 degrees and west longitudes negative. This gives a possible range of longitudes of -180 degrees to +180 degrees. The launch site that is used is south and east of the ionospheric point, with latitude '40' and longitude '-80'. This results in Screen 11.

Screen 11

```
Enter launch pt. latitude           (      .000000000     ):40
Enter launch pt. longitude          (      .000000000     ):-80
```

Following this, item '5' should be selected from the Launch Parameters menu. The purpose of this item is to provide the third component of the location of the ray launch site, namely the starting height. The starting height is given in kilometers above the earth's surface. This exercise will start on the earth's surface, so the response that is needed is to simply type return so that the default value shown is accepted. Before hitting return, the screen will look like Screen 12.

Screen 12

```
Enter starting ht. (km)                (      .000000000     ):
```

Upon return to the menu, select item '6'. This set of prompts is used by the program to obtain from the user values which will determine when the problem should be terminated. The first parameter is the range limit. This represents great circle range on the earth's surface. Internally this is dealt with as an angular range, but it is easier to think in terms of ground range when entering the value. This must not be confused with the distance traveled along the raypath! The distance along the ray is, in general, longer than the associated ground range. The value selected for this example is '2000' km. The second value to be entered is the altitude cutoff. This gives the height above the surface of the earth of a spherical shell which, when pierced, causes the raytracing to terminate. For this example a reasonable value is '400' km. The resulting screen is Screen 13.

Screen 13

```
Enter range limit (km)                 (      3000.00000     ):2000
Enter altitude limit (km)              (      36000.0000     ):400
```

The next quantity to be entered is the raypath increment, selection '7'. This value determines how finely the raypath is broken up for the purposes of the calculation. There is an interplay here of the desire to make the path exceedingly fine for the sake of accuracy, with the desire that the raytracing complete in a finite (and preferably small) amount of time. The appropriate range of values has already been discussed. It is sufficient that the default value of '4' km

represents somewhat of a balance and so this is the value chosen. The prompt is displayed in Screen 14.

<u>Screen 14</u>

```
Enter raypath increment (km)            (       4.00000000     ):
```

The final important selection from the Launch Parameters menu is item number '8', the wave frequency. This is the value of the frequency of the radio wave that the ray represents. The appropriate range of values for this is discussed above in the operating instructions section. For the purposes of this tutorial, it is desired that the ray not penetrate the ionosphere, so a wave frequency of '4' MHz is selected. The default of five MHz is merely a placeholder so that the program doesn't accidently start with frequency equal to zero and promptly crash due to division by zero. The screen containing this prompt is represented by Screen 15.

<u>Screen 15</u>

```
Enter wave frequency (MHz)              (       5.00000000     ):4
```

After completing this, all the necessary data has been entered in the Launch Parameters menu. The user may therefore enter '9', the selection indicating that the data entry is finished in this menu. The control of the program is returned to the main menu. From the main menu, the other data entry menu may now be entered. This is done by making selection '2' from the main menu. The Elevation Parameters menu now comes on screen.

As with the Launch Parameters menu, this tutorial will now proceed through the items in the Elevation Parameters menu. The first item in this menu deals with the array of azimuth and elevation values for which rays will be sent out. Select '1' as in Screen 16.

Screen 16

ELEVATION PARAMETERS MENU

```
    1 - Dimension of problem

    2 - Starting azimuth

    3 - Starting elevation

    4 - Elevation resolution

    5 - Azimuth resolution

    6 - Elevation limit

    7 - Azimuth limit

    8 - Done

Your choice (1-8 only, please)?          (    0):1
 Press RETURN to continue.               (          ):
```

This selection will provide a prompt asking as to the dimension of the problem. This is defined earlier, however a brief rundown is in order. The term dimension refers to the dimension of the array that could be used to represent the elevation and azimuth values to be used. For this reason, dimension 0 represents just a single value of elevation and azimuth. Dimension 1 represents a single azimuth value and multiple elevation values. Finally, dimension 2 corresponds to the case where there are multiple elevations and azimuths to be used. So that all the menu items may be selected, a value of '2' should be entered here. This is represented by Screen 17.

Screen 17

```
Enter dimension of problem               (       .000000000     ):2
```

As usual, entering the response above returns the user to the menu. Proceeding down the menu in order, the next selection is '2'. This item allows the entry of the initial azimuth value for the problem. The azimuth is given in degrees and is taken relative to geographic north in the usual manner. Because this example involves only a spherically symmetric ionosphere, one direction is as good as another. The default value of '0' is therefore taken, by typing a return. This situation is depicted in Screen 18.

Screen 18

```
Enter starting azimuth (deg)            (        .000000000     ):
```

Item three of the Elevation Parameters menu is the next one selected. This prompts for the starting elevation angle. Elevation is measured in degrees, with the zenith as +90 degrees, and the nadir as -90 degrees. Negative values of elevation angle come into play for cases when the launch point of the ray is at a non-zero altitude. For this example, a reasonable value to use is '10'. Screen 19 shows this.

Screen 19

```
Enter starting elevation (deg), use neg.
for values >90 deg from zenith.         (        .000000000     ):10
```

The fourth selection from the Elevation Parameters menu deals with the elevation separation between rays. Very little coaching can be given about appropriate values for this parameter. The fineness or coarseness of the elevation resolution must be determined in terms of what is appropriate for a given problem. Since this problem is of an instructional nature, an arbitrary value of '0.5' degrees has been chosen. This is shown in Screen 20.

Screen 20

```
Enter elev. resolution (deg)             (       .000000000    ):0.5
```

Next in line is the fifth menu item of the Elevation Parameters menu. The azimuth spacing between rays is entered at the resulting prompt. This is handled in a manner identical to that of the previous prompt. The value which is to be entered is the same as that for elevation, namely '0.5'. Screen 21 shows this.

Screen 21

```
Enter azimuth resolution (deg)           (       .000000000    ):0.5
```

The method of specification for the elevation and azimuth values depends upon having the starting value, the division size, and the ending value. The ending value for elevation is given by selection of item six from the Elevation Parameters menu. Internally, the difference between the start value and the limit is taken. The number of increments of size equal to the resolution value that will fit into this interval is then calculated. This is the number of elevation steps that will be taken. For the sake of completing the example problem quickly, the limit will be taken to be '10.5' so that rays will be sent out at only two elevations. The response is shown in Screen 22.

Screen 22

```
Enter elev. limit (deg)                  (       .000000000    ):10.5
```

Lastly, menu item seven is selected. This allows the azimuth limit to be entered in a manner identical to the elevation limit. Again, the value is chosen so that only two values of azimuth will be used. The combination of these elevation and azimuth limits result in just four rays being calculated. The appropriate value to use for the azimuth limit is then '0.5'. It should be remembered in setting the both of the limits that if a specific number of rays is desired in elevation and in azimuth, that the end points must be counted properly. The prompt and response are shown in Screen 23.

Screen 23

```
Enter azimuthal limit (deg)              (       .000000000    ):0.5
```

All of the necessary parameter values have been entered now. Selection of item '8' of the Elevation Parameters menu causes a return to the main menu. Everything is now ready for the raytracing to be performed. Therefore, item '3' is now selected from the main menu. Because this is a fresh problem, no prompting about the disposition of the entered parameters is necessary. These values are automatically written out to the RAYDAT-type file that was specified at the beginning of this exercise, namely TRIAL.DAT. The user is notified that the ray-tracing has begun. When the calculations are complete, the user is notified and the file to which the results have been written is also noted. The user is then given the opportunity to do another raytracing problem. In order to give a feel for what the program will do, the

answer to this prompt is 'y' for yes. The screen should look like that shown in Screen 24.

Screen 24

```
BEGINNING RAY LOOPS.

RAY LOOPS DONE.


Results written to file FOR040.DAT

Do another problem (Y/N)?                (            ):y
```

The affirmative response causes RAYTRACE to return to the main menu. For the second problem, just a single ray will be traced. To do this, select '2' from the main menu in order to proceed to the Elevation Parameters menu. Once at the Elevation Parameters menu, choose '1' to change the current setting for the dimension of the problem. Recall that a problem of dimension zero is a single ray problem. The prompt that comes on the screen for the dimension of the problem contains the current value, which is 2. Enter a value of '0'. This is shown in Screen 25.

Screen 25

```
Enter dimension of problem          (      2.00000000     ):0
```

It is important to note that no other values in the Elevation Parameters menu need be changed. In fact, it will be very instructive to attempt to change any of the values corresponding to menu items 4-7. The program will not allow these items to be changed, instead the user is returned to the menu. These items are irrelevant to the calculation once the dimension has been set to zero. Because the purpose of this second portion of the problem is simply to do a second portion to the problem, this one change will suffice. Therefore, select '8' to return to the main menu, and then select '3' to proceed with the raytracing.

This time, before the raytracing is started, RAYTRACE notes the possibility that the user has made changes to parameter values. The user is asked whether or not to update the data file. If the answer is yes, the current values are saved out to the RAYDAT-type file. If the answer is no, then if the user chooses to end the session after the current raytracing, the changed parameter values disappear. For the purposes of this example, it doesn't matter whether the values are saved or not: the response of 'n' has been chosen. This may be seen in Screen 26.

Screen 26

```
Update the datafile (Y/N)?                 (          ):n
```

The program then signals that the tracing of the ray has begun. When the end of the calculation is reached, the display is similar to that seen before. This time, however, it is desired to end the problem. The response to the prompt is therefore 'n'. The screen will look like Screen 27.

Screen 27

```
Update the datafile (Y/N)?             (          ):n

 BEGINNING RAY LOOPS.

 RAY LOOPS DONE.


Results written to file FOR041.DAT

Do another problem (Y/N)?              (          ):n
 Press RETURN to continue.             (          ):
```

The final thing that the program does is to display a message. This message is to warn the user to take proper care of any data files that have been produced during the raytracing session. The point cannot be overemphasized that the filenames given are far from descriptive and results may get lost or destroyed if a myriad of such files exists. The closing message is reproduced in Screen 28.

Screen 28

```
Session done.  For safety's sake, copy the data
and result files into a separate directory to
prevent accidental overwriting.
```

At this point, RAYTRACE has finished execution and control has returned to whatever system is being used. Several new files should now be in existence. TRIAL.DAT is the file of the RAYDAT type that contains the saved parameter settings and the ionospheric specifica-

tion. A grid file called GRID.DAT was created by entering the ionospheric data from the console. There are also two results files, FOR040.DAT and FOR041.DAT. As the final message warns, these files should either be moved or renamed. Examining the results files is discussed in a later section.

New session using data file input

This second example will build upon the foundation of the first. The grid file that will be used for input is going to be the GRID.DAT that was produced in the previous example. Reference will also be made to the first example for those portions of the two examples which are identical. It is expected that many RAYTRACE runs will be similar to the current example in that most fresh problems will be initiated by use of an input file specifying the ionosphere.

This example begins by starting RAYTRACE as before. The initial prompt comes up on the console. Since this example is to be a new problem, respond with '1'. The prompt asking for the name of the file to store the ionospheric specification and input parameters now appears. Recall that last time the file TRIAL.DAT was used. If this file hasn't been relocated or renamed, entering this name again may cause a problem. So that the files may be distinguished the name that is chosen for this example is 'TRIAL2.DAT'. After this response, the screen should look like Screen 29.

<u>Screen 29</u>

```
Is this a new (1) or old (0) problem?   (    0):1

Enter filename for storage of ionospheric
information (10 char max):              (RAYDAT.DAT):trial2.dat
```

The next prompt is the one asking for the file containing the ionospheric specification. Unlike the first example, one now exists. The file even has the name GRID.DAT. This is the file to be used, so

the default value of the prompt may be accepted by typing a return. This situation is depicted in Screen 30.

Screen 30

```
The ionospheric grid file name is?
 (type NONE if none exists)              (GRID.DAT  ):
```

The user should now notice a difference from the previous example. Because the ionospheric specification already exists, the program has no need to prompt the user for keyboard entry of the ionospheric parameters. RAYTRACE immediately proceeds to the main menu. From here the previous example may be picked up again. Menu item '1' from the main menu should be chosen. The user should now enter values for each of the items in the Launch Parameters menu. The values to be used are just those of the previous example. That example and its discussion may be followed and will not be repeated here.

Following completion of the Launch Parameters menu, the next step is to proceed to the Elevation Parameters menu by selecting '2' from the main menu. Again, the items of this menu will be completed in numerical order. Just as in the first example, enter '1' as the menu choice. For the sake of speed, just a single ray is desired. Therefore, the response to the resulting prompt will be to accept the default value by typing a return. This is shown in Screen 31.

Screen 31

```
Enter dimension of problem               (        .000000000     ):
```

Next, choose item '2' from the Elevation Parameters menu. The value is chosen to be the same as in the previous example. Recall that this means the azimuth value will be 0 degrees. The default response is therefore to be accepted. Screen 32 show this.

Screen 32

```
Enter starting azimuth (deg)              (       .000000000     ):
```

The starting elevation value is the next to be set. Menu item '3' should be selected. At the resulting prompt, a value of '10' degrees should be entered. This ought to lead to a ray with identical results as one of those done in the first example. The results may be compared as a check. The elevation data entry is shown in Screen 33.

Screen 33

```
Enter starting elevation (deg), use neg.
for values >90 deg from zenith.           (       .000000000     ):10
```

Because the dimension of the problem has been selected to be zero, there is no further data entry needed. Item '8' should therefore be selected from the Elevation Parameters menu. Upon return to the main menu, the user may choose item '3' as in the previous example. This will initiate the raytracing without any intervening prompts. Because this is a fresh problem, the data is automatically recorded in the RAYDAT-type file specified at the start. When the raytracing is complete, the program behaves in a manner exactly identical to that shown in the first example. For the purpose of practice, answer the prompt for another problem with 'y'. This is shown in Screen 34.

Screen 34

```
BEGINNING RAY LOOPS.

RAY LOOPS DONE.


Results written to file FOR040.DAT

Do another problem (Y/N)?                  (          ):y
```

The program returns to the main menu for another time through the problem. So that the problem will be somewhat different, a new elevation value will be used. To achieve this, select '2' from the main menu, and then '3' from the Elevation Parameters menu. An elevation angle of '8' degrees will be used. This will result in a screen that looks like Screen 35.

Screen 35

```
Enter starting elevation (deg), use neg.
for values >90 deg from zenith.          (      10.0000000    ):8
```

This is the only change that is necessary, so choose '8' from the Elevation Parameters menu. When control has returned to the main menu, selection '3' may be made so that the calculation may begin. Again, as with the first example, the query about updating the data file is seen. There is no need to preserve the change that has been made, so answer with 'n'. The raytracing then proceeds as usual, displaying the normal information. Because this example is essentially complete, respond with 'n' to the prompt asking about doing another problem. This is

shown in Screen 36 below. RAYTRACE then displays its final message and returns to the system.

Screen 36

```
Update the datafile (Y/N)?            (          ):n

 BEGINNING RAY LOOPS.

 RAY LOOPS DONE.


Results written to file FOR041.DAT

Do another problem (Y/N)?             (          ):n
 Press RETURN to continue.            (          ):
```

Old (saved) session

This third example illustrates the use of a RAYDAT-type file for input at the beginning of the problem. As with the second example, this example will be built upon the previous ones, and only those portions which are new for this example will get detailed attention. The type of session modeled by this example is representative of perhaps the majority of RAYTRACE sessions. By reading in a RAYDAT-type file, all of the parameters have some pre-existing setting. Only those parameters that are different for the particular problem at hand need be examined and changed.

First, start RAYTRACE in the usual manner. This time, however, at the initial prompt, the response will be '0' to indicate that this is an old problem that will initially use the parameter values stored in a RAYDAT-type file. The second prompt is the same as has been seen before. This time its purpose is to get the name of the file from which the parameters will be read. Any parameter changes may be stored

in this file also. For this example, the file created by the first example, 'TRIAL.DAT', will be used. This is shown in Screen 37.

Screen 37

```
Is this a new (1) or old (0) problem?   (    0):0

Enter filename for storage of ionospheric
information (10 char max):               (RAYDAT.DAT):trial.dat
```

RAYTRACE now continues directly to the main menu. There is no need to read from a grid file because the pertinent data are stored in the RAYDAT-type file along with the parameters. At this point, if the parameters desired are the ones that were saved during the last session, the user could just select the third item from the main menu and proceed directly with the raytracing. For this example, however, recall from the first example that the parameters entered included a dimension of two for the problem. In the interests of speeding the example along, this should be changed to a dimension of zero. To do this, the user should select '2' from the main menu to reach the Elevation Parameters menu. Then '1' should be selected to reach the prompt from which the dimension may be set. Notice that the default value given is indeed two. Enter the new value of '0' as shown in Screen 38.

Screen 38

```
Enter dimension of problem          (      2.00000000     ):0
```

This is the only change that needs to be made, so enter a choice of '8' to return to the main menu. Now select '3' from the main menu to begin the raytracing. The program now presents prompts that have

not been seen until now. Because the parameters may have been changed, even though this is the first time through the problem, the user needs to be presented with the opportunity to preserve these changes. First, the user is asked if the values should be saved in the current file. If they are not to be saved there, the user is then asked if a new data file should be used to backup the new values. A negative answer to both of these prompts will mean that if the problem is ended the altered parameters will not be saved anywhere. It is not necessary that the change that has been made be saved, so answer 'n' to both prompts.

Some additional comments about these prompts are in order. If the old parameters are overwritten, then the data are safe and there is no need for the second prompt. RAYTRACE therefore skips it and goes on with the calculation. If a backup is desired, the user will be prompted for the filename to be used. Also, on succeeding times through the raytracing the prompt asking about updating the data file (as in Screen 36 above) will be referring to the last file that was written to.

After dealing with the above mentioned prompts, the raytracing is performed. The rest of the prompts are just as they have been before. There is really nothing further to be discovered, so 'n' may be entered in response to the query about another problem, as shown in Screen 39. After this the program ends in the usual fashion, returning to the system. This concludes the final example.

Screen 39

```
Overwrite existing file (Y/N)?          (            ):n
Backup the datafile (Y/N)?              (            ):n

 BEGINNING RAY LOOPS.

 RAY LOOPS DONE.


Results written to file FOR040.DAT

Do another problem (Y/N)?               (            ):n
 Press RETURN to continue.              (            ):
```

Remarks

The three examples given serve to illustrate essentially every feature of RAYTRACE. After following all three examples, the user should be familiar enough with the interface of the program to go ahead and "play around" by trying different things. Such "play" may be worth the time if the user feels the need for additional familiarity with the operation of the program. The menu reference sheet given later should also be of some help.

EXAMINING RESULTS

So far, the discussion has dealt with the use of the RAYTRACE program. The program is of very little use, though, unless there is some way of getting at the results from the raytracing runs. This section deals with the program READER2 that is provided with RAYTRACE.

READER2 represents a zeroth-order program for examining the results of RAYTRACE. While this program has proven adequate in the testing and development of RAYTRACE, it is certain that the end users of RAYTRACE will need to write their own, more sophisticated analysis software.

READER2 is very straightforward to use. When the user executes it, the program prompts for the name of the results file that it is to be read. Upon receiving the name, READER2 immediately displays the important values from the result file. This display is done for each pair of elevation and azimuth values. Results for each data point on the ray are displayed with the values for a particular point taking one screen. Recall that data points on the ray are earth impact points and the end point of the ray. After display of each data point, READER2 pauses and awaits the user's response before continuing. When all the data are exhausted, the program simply ends and returns to the system.

The values displayed on the screen are by and large self-explanatory. The elevation and azimuth values are given in degrees. The direction cosines give the orientation and direction of the ray at that particular point in an SEZ coordinate system where X is South, Y is East, and Z is up. The deviation value represents the deviation away from where the ray would have landed had it stayed on a great circle path. This is a measure of the cross-path tilt of the ionosphere.

Several comments on the regular usage of READER2 are in order here. Although the display of the results on screen is useful, it is often more useful to have a hardcopy of these results. This is achieved not by READER2, but by the system that it is operating on and the console used. Many systems and consoles have the capability to either redirect output to a disk file, or perform screen dumps to a printer. Given the rather widespread presence of such a capability, no provision has been made to produce a printable file. Since the user is likely to develop individualized analysis software anyway, there is really no need for this capability.

Another comment concerns the blank record that will occasionally be displayed. This is entirely harmless. It occurs because provision must be made for the possibility that the limit on the number of bounces may be reached and there still needs to be the additional space for the end of ray point's data. If the bounce limit isn't reached, then this extra space is blank.

TIPS, RESTRICTIONS, REMINDERS, ETC.

This section has the purpose of gathering together some of the comments that have either gone unsaid to this point, or bear repeating. First, there are some comments of a general nature. Comments specific to a particular implementation or environment follow. Finally, some usage hints are presented.

General comments

The speed with which RAYTRACE operates will be of some concern to many users. Some benchmarks have been run on various machines. The two primary implementations discussed in this User's Guide are the VAX/VMS and the DOS versions. The machines compared were an IBM AT-compatible (with math coprocessor) and a VAX 11/785. The benchmarked run uses the TRIAL.DAT file from the four ray example of the tutorial examples above. The run was repeatedly made and the results averaged. Timing was done by stopwatch from typing return at the last prompt before the execution of the ray loops until the message of "RAY LOOPS DONE" appeared on the screen. For the VAX, in addition to the apparent execution time as measured by stopwatch, the CPU time was taken by using the <CTRL>-T mechanism of VMS for obtaining the process time statistics. Because the VAX is a multiuser machine, the usage level is of some significance. The benchmark was run early in the morning before the many users were on the system. At the time of the runs, there were only four users, so that the machine could be termed lightly loaded.

The results of the benchmark are as follows:

Table 2

Machine		Time	
AT-clone	:	40.7	sec
VAX 11/785	:	9.2	sec (user's time)
VAX 11/785	:	5.1	sec (CPU time)

Due to the usage level on the VAX when the benchmark was run, it is fairly clear that the user time could have been somewhat better, but it also could have been much worse. Extrapolating from other experience, the execution time for this benchmark on any of the current (1987-88) crop of 32-bit microcomputers would lie somewhere midway between the VAX's user time and CPU time.

Several restrictions of RAYTRACE should be noted. Filenames are to be ten characters or less in length. The input files need to be located in the same directory as the executable for RAYTRACE, at least the way RAYTRACE is currently written. Another restriction is on the size of the input data. Currently, the grid of points cannot be larger than 30 in latitude by 60 in longitude. This trims the size of the memory allocated to a level tolerable to the DOS specifications stated previously. A third restriction is that rays may not be fired with an elevation angle of 90 degrees. This restriction only applies if the ray will make a reflection from the ionosphere. Rays which penetrate the ionosphere may have elevations of +/- 90 degrees.

It is important to make sure when doing a new problem that ALL of the menu items have been selected and checked to verify that their values are properly set. Failure to do so is perhaps one of the most frustrating of mistakes that can be made while using RAYTRACE, as the program is perfectly willing to use whatever the current values are even though they may not be what the user intends. As a further reminder, the following table lists some parameters and their range of values.

Table 3

Parameter	Range
Latitude	+90 to -90 deg with (+) north
Longitude	-180 to +180 deg with (-) for west longitudes
Plasma frequency squared	Megahertz squared
Heights and distances	Kilometers
Elevation angles	+90 (zenith) to -90 (nadir) degrees
Azimuth angles	0 to 360 degrees with 0 north and values increasing eastward

Some of the parameters that the user enters have either imposed or practical bounds. The wave frequency of the RF transmission that the ray represents is bounded on the lower side to a few MHz. This is because for lower frequencies magnetic field effects, which are not included in the current version of RAYTRACE, play an increasing role. As an upper bound, frequencies beyond a few GHz begin to lose reliability for two reasons. First, as a calculational matter, the ratio of the plasma frequency squared to the wave frequency squared becomes very small. Operations occur of the type $(1 - x)$ where x is a small number, with the resulting loss of accuracy. Second, refraction due to the neutral atmosphere becomes a much more important factor at these frequencies. These limits are then limits on the believability of the results. RAYTRACE will calculate over a much larger range of frequencies, but the results will be far from reliable.

The size of the raypath increment is another quantity that has practical limits. There is a lower limit due to accumulation of error and the duration of the calculation. For raypath increments smaller than a few tenths of kilometers, even modest calculations begin to take intolerable lengths of time. This conclusion is of course subjective and depends upon the computer being used at the time. The problem of duration of the calculation has, however, prevented rigorous testing for the degradation of the results due to accumulation of errors. An upper limit also exists on the size of the raypath increment. Increments which are too large begin to violate an approximation made in the theory underlying the raytracing algorithm. Typically, increment sizes of 10-15 km are about the maximum practical. As with the limits on the wave frequency, values outside these may be used, but the results must be viewed as ranging in quality from rough approximations to completely unreliable.

The cutoff height is a quantity that bears further discussion. The default value initially given is essentially the height of geosynchronous orbit. This value represents a compromise. The ionosphere is considered to end at a height of 2000 km in the model used. Propagation to altitudes beyond this is modeled as straight line propagation through free space. While this is certainly not rigorously correct, it is by and large an accurate assumption. Clearly some cutoff must be

used so that the program doesn't calculate off to infinity. Since most radio operations currently take place within a sphere defined by the radius of geosynchronous orbit the default value seems convenient. The value may be reset to any value the user desires. Be warned, however, that the neglected refraction will eventually add up as the path length increases. Depending upon the application, this error could have a substantial effect.

VMS specific comments

The points that are mentioned here have been discussed in greater detail elsewhere and are referred to here as a reminder. When transporting the source code between a DOS-based and a VAX/VMS system, there are a few lines that may need to be changed. These changes are discussed in detail in the Setup section. The naming and renaming of files needs to be carefully considered. The number of results files having similar names but different version numbers can become bewildering. Also, the user should automate as much of the raytracing process as possible through the use of command files.

When using VMS command files to perform batch RAYTRACE jobs, one drawback becomes obvious, there is no provision in such command files for a response that involves just typing a return. Every line of the command file must contain something. For this reason, any prompt of the 'Press RETURN to continue.' type will accept '.' and return as if it were just a plain return. Therefore, on any line of a command file where the desired response is just a return, a line with a single period on it will serve as the necessary substitute.

DOS specific comments

The points that are mentioned here have been discussed in greater detail elsewhere and are referred to here as a reminder. When transporting the source code between a VAX/VMS system and a DOS-based system, there are a few lines that may need to be changed. These changes are discussed in detail in the Setup section. The naming and renaming of files needs to be carefully considered. Because DOS

provides little protection against accidental destruction of files, the user must exercise extreme care to preserve files against disaster. Also, the user should automate as much of the raytracing process as possible through the use of batch files.

Uses and Usage tips

RAYTRACE is a tool that may be used to accomplish a number of tasks in the realm of propagation modeling. It may not be immediately clear, though, how to do some things with the program. The following is a discussion of a few sample uses.

The instructions for operating RAYTRACE discussed the concept of the dimensionality of the problem. From that discussion, it should be clear how to trace single rays, vertical fans of rays, and arrays of rays. A variation on this theme that might prove useful is the horizontal fan of rays. This may be achieved by using a two dimensional problem with only one allowed elevation angle. The mechanics of doing this involve setting the starting and ending elevation values equal (or nearly equal) and entering a step size that is arbitrary (or large compared to the difference between the starting and ending elevations).

Information on the coverage of an area by a broadcast transmitter may be obtained by tracing a two dimensional set of rays. It should be noted that the resulting array of landing points for these rays will not be evenly spaced in latitude and longitude. It is possible, though, to interpolate on such a grid and produce contours or other desired information. At present the only way to achieve (approximately) the desired spacing between points is by experimentation with a small number of rays.

Tracing rays between two specific points (homing) is also a desirable capability. While such a capability is not currently built into RAYTRACE, it is relatively straightforward to do manually. By tracing a pair of rays that are fairly close in elevation, the results for the end points of the pair of rays may be used in a linear interpolation scheme. Typically it requires only five or so iterations to reach the desired convergence on the specific target point. A similar process may be done for the azimuth, although the azimuth is usually

fairly well approximated by that of the great circle path between the two points. An azimuth search will usually also converge more quickly.

It is sometimes desirable to have detailed information along the raypath. There currently exists some code within RAYTRACE to report the height and range at each increment along the ray, for the purpose of later plotting of the raypath. This code is normally commented out, as the files of coordinates produced may become enormous. This code is located in the routine RAYSUB a few lines below line 30300 (consult the source code listing, Appendix D). It merely consists of a WRITE statement and its corresponding FORMAT statement. Uncommenting these lines and recompiling will yield a version of RAYTRACE that will produce an extra generic data file with pairs of values: ground range, height. These values may then be used in a plot of the trajectory of the ray. Certainly, more sophisticated schemes are possible, but such things are best tailored to a given need.

Ultimately, RAYTRACE is extensible. This is most easily achieved by extending the menu system. Adding to the menu system allows more parameters or flags to be entered by the user. Flag values entered in this manner could be used to activate or deactivate additional sections of code. This code could then operate upon existing and new parameters for the calculation of additional results. Details of adjusting the menu system are given in a comment block in the routine DOxDATA, just after line 12002. The menu files are discussed in Appendix A. The only other requirement is the addition of the new features or calculations at the appropriate locations in the existing program. This may only be determined by study of both RAYTRACE and the theoretical groundwork for the intended addition.

MENU QUICK REFERENCE

The three menus of RAYTRACE are presented here, with brief explanations of most menu items. They are given one per page, so that the user can easily photocopy them for handy reference. There is also ample room for notes. One suggestion, if a copy machine that reduces is available, is to copy the three menus reduced, then paste them on one sheet. Copying this sheet will give a single page "cheat sheet".

MAIN MENU

```
1 - Edit launch parameters            (Go to Launch Parameters menu)

2 - Edit elevation/azimuth parameters  (Go to Elevation Parameters
                                                    menu)
3 - Proceed with raytracing

4 - Quit program
```

LAUNCH PARAMETERS MENU

1 - Bounce limit	(limit number of earth impacts)
2 - Signal intensity	(flag to turn on/off intensity loss calculation)
3 - Conductivity	(select type of earth surface for reflection loss)
4 - Launch point	(latitude/longitude location of start point)
5 - Launch height	(altitude of start point)
6 - Range & height limits	(set great circle ground range and altitude cutoffs)
7 - Ray path increment	(set step size along the ray)
8 - Wave frequency	(frequency of transmission modeled by the ray)
9 - Done	(return to the main menu)

ELEVATION PARAMETERS MENU

1 - Dimension of problem	(determine multi-ray launch pattern)
2 - Starting azimuth	(set initial launch azimuth)
3 - Starting elevation	(set initial launch elevation)
4 - Elevation resolution	(set elevation step size for multi-ray calculations)
5 - Azimuth resolution	(set azimuth step size for multi-ray calculations)
6 - Elevation limit	(set limiting value on elevation for multi-ray calculations)
7 - Azimuth limit	(set limiting value on azimuth for multi-ray calculations)
8 - Done	(return to main menu)

APPENDICES

Appendix A:
File Formats

GRID.DAT

The grid file is a means by which ionospheric information may be input into RAYTRACE. The grid file is generally produced externally to RAYTRACE, either by use of an ionospheric model program or by processing of actual ionospheric data. The file contains the pertinent ionospheric information at a gridded set of latitudes and longitudes. Grids may be entered from the console, but this is only practical for relatively small grids.

Grid files are composed of a number of data arrays. First, the definition of the latitude and longitude grid is recorded. Following this, the six ionospheric parameters required for specification of the vertical profile are given for each point on the grid. The six parameters are the maximum plasma frequencies for the E, F1, and F2 layers, the heights for the maximum frequencies for the F1 and F2 layers, and the semithickness of the F2 layer. The choice of these parameters is determined by the profile model that is built into the current version of RAYTRACE. Finally, the date, time, and sunspot number are recorded.

The array of values that serves to define the latitude and longitude grid is a real array containing six parameters. They are the latitude and longitude spacing for the grid, followed by the initial latitude and longitude values for the grid. The starting point is taken as the southwest corner of the grid. Values are in degrees and longitude is given as east of Greenwich positive, west negative. This gives longitudes running from -180 to +180 degrees. The last two values in the array are the number of points that the grid is to contain in the latitude direction and in the longitude direction respectively. The grid must be rectangular and if necessary should be either padded with more data (if the grid is being generated from a model or interpolation

of actual data) or some points eliminated so that it becomes rectangular.

The arrays of ionospheric data are written as two dimensional arrays with the first index corresponding to latitude and the second to longitude. The indices represent the integer count of grid points traversed in the given direction to reach the coordinates of the grid point. The first array contains the peak plasma frequency squared values for the E-layer, for the entire grid. The second array holds the height of the peak of the F1-layer and the third holds the plasma frequency squared values for the F1 maximum. The last three arrays contain the values which are used to specify the F2-layer. They are, in order of appearance, the semithickness, the height of the layer maximum, and the plasma frequency squared at the maximum.

The final array is an integer array containing the time, date, and sunspot number for which the problem is being run. These values are contained in the array in this order: sunspot number, year, month, day, hour (UT), and minute. The current configuration of RAYTRACE no longer uses this information directly as the portions of code that use it have been commented out. They have been left in both for the possibility of future calculational use, and so that the information may be included in displays or reports if desired.

The format of the GRID.DAT type of data file is perhaps best illustrated by including fragments of code that RAYTRACE uses to read information from the grid file. The fragmented listing (Listing 1) includes the declaration statements. The file is written unformatted to save time and space. The filename is set either as a default value, or from a prompt to the user. The correspondence between the arrays mentioned above and those in Listing 1 is fairly obvious. The variables are named according to the convention that is presented with the source code listing in Appendix D.

Listing 1

```
      INTEGER IVxPARAM(6)
C
      REAL*8 RVxGRID(6)
      REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
      REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
C
      CHARACTER*10 KVLxFIL
                         *
                         *
                         *
      OPEN(20, FILE=KVLxFIL,FORM='UNFORMATTED')
      REWIND (20)
      READ(20) RVxGRID
      READ(20) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) IVxPARAM
      CLOSE(20)
```

RAYDAT.DAT

The RAYDAT.DAT type of file is used by RAYTRACE for the storage and retrieval of both the ionospheric information and parameters of the problem that the user has entered. For the sake of simplicity, the ionospheric information is stored in exactly the same manner as in the case of GRID.DAT type files. Arrays containing the other problem parameters are then added in after the ionospheric values. As a consequence of this file structure, only the arrays that are not repeated from GRID.DAT need be discussed.

The first of these arrays is an integer array containing three values. The bounce limit is stored first. This sets a bound on the number of times that a ray may impact the earth, which is useful in some cases dealing with subionospheric raypaths. Next is a flag that determines whether or not signal loss calculations will be performed.

Finally, there is a flag that determines what sort of surface will be used in reflection loss calculations for earth impacts of the ray.

The second array is a real array containing the latitude and longitude of the launch point. This is followed in the array by the range limit. This parameter sets a bound on the great circle ground range that the ray is allowed to propagate to. The fourth value in the array is the size of the raypath increment. Fifth is the wave frequency. The sixth value is the starting height of the ray, above the earth's surface. Finally, the seventh value is the altitude cutoff value.

The final array is a real array that contains the information necessary to specify the elevations and azimuths at which rays will be sent out. The first parameter is the dimensionality of the problem. This is defined by considering the aim points of the ray in an elevation versus azimuth plane. A single ray is a single point, and so has dimension zero. A fan of rays having a particular azimuth forms a line of points, and is considered dimension one. Lastly, a set of such fans produces an array of points on the plane and is therefore considered as dimension two. After the dimensionality of the problem comes the initial azimuth and the initial elevation. Following these values, the azimuth and elevation spacing between rays is stored. Last of all, the final azimuth and elevation values are stored.

As above, the easiest way to make the structure of the data file manifest is to present the fragments of code responsible for writing to the file. The code is presented in Listing 2. Again, the filename is set previously to its proper value. The additional arrays are IVxLAUN, RVxLAUN, and RVxOPTION.

Listing 2

```
      INTEGER IVxPARAM(6), IVxLAUN(3)
C
      REAL*8 RVxGRID(6), RVxLAUN(7)
      REAL*8 RVxOPTION(7)
      REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
      REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
C
      CHARACTER*10 KVxDFIL
                          *
                          *
                          *
      OPEN (30, FILE = KVxDFIL, FORM='UNFORMATTED')
      REWIND (30)
      WRITE (30) RVxGRID
      WRITE (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) IVxPARAM
      WRITE (30) IVxLAUN
      WRITE (30) RVxLAUN
      WRITE (30) RVxOPTION
      CLOSE(30)
```

FOR###.DAT

This file is the results file for RAYTRACE. The ### represents a number that is assigned sequentially during the course of a given ray-tracing session so that the different runs during the session may be distinguished. Because results files may need to be of greater portability, the results are written to disk in formatted form. The results are stored by first saving the information necessary to determine the number of data points represented in the file. Data points consist of earth bounce points, and the terminal point on the ray. The azimuth and elevation of a particular ray are saved and then the array

containing the information from all the data points on that ray is stored. This then repeats until the data from all rays has been saved.

The number of data points per ray is determined from the maximum number of bounces allowed by the bounce limit, plus one for the terminal point of the ray. The proper number of rays are then accessed by using the number of azimuths and the number of elevations recorded. These set the limits on two loops, within which the file is accessed.

The array of data for a given ray is two dimensional. The first index is the number of the data point along the ray. The second represents the particular data element for that point. The meaning of these elements is given in a comment block in the source code for the subroutine RAYSUB, just after the variable declarations. The ones that are currently used are as follows. First are the latitude and the longitude. The third element is the group time delay along the ray to that point. The great circle ground range is next, followed by the signal loss in dB, due to spatial loss, focusing or defocusing, and possibly reflection loss. The sixth element contains the amount that the ray has deviated from a great circle path at that point. The ninth element of the array gives the height of the ray at the data point. Element ten contains the phase path length along the ray to that point. The next three elements give the x, y, and z direction cosines of the ray at that point. Here, the x, y, and z directions are in the coordinate system where x is south, y is east, and z is up.

The sections of code for writing the results out to the file have been excerpted from the main routine of RAYTRACE. The code is presented in Listing 3. Also presented is the code that produces the number associated with the file. The integer IVxIND is initialized to 40 and incremented every time a particular run is complete. It is then used to produce the filename of the results file.

<u>Listing 3</u>

```
      INTEGER IVxIND, IVxLAUN(3), IVxAZLIM, IVxELLIM
C
      REAL*8 RVxAZI, RVxELEV, RVxBOUN(11,15)
C
      CHARACTER*2 KVxIND
      CHARACTER*10 KVxOFIL

                        *
                        *
                        *

      WRITE (KVxIND,'(I2)') IVxIND
      KVxOFIL = 'FORO'//KVxIND//'.DAT'

                        *
                        *
                        *

      OPEN (IVxIND,FILE = KVxOFIL)
      WRITE (IVxIND, 22000) IVxLAUN(1)+1, IVxAZLIM, IVxELLIM

                        *
                        *
                        *

      WRITE(IVxIND, 21003) RVxAZI, RVxELEV

                        *
                        *
                        *

      WRITE(IVxIND, 21003) ((RVxBOUN(L,K),K=1,15), L=1,IVxLAUN(1)+1)

                        *
                        *
                        *

      CLOSE(IVxIND)
21003 FORMAT (G24.14)
22000 FORMAT (I4)
```

MENU FILES

The scheme for using and presenting the menus and menu files is detailed in the source code listing in subroutine IOxMENU. The method is reviewed here. The file MASTER.MEN is used to maintain a list of the rest of the menu files. It is merely a text file that contains the names of the other menu files, with each filename on a separate line. RAYTRACE is currently set up to handle up to ten menu files in addition to the master file. The filenames are ordered according to their numbering within RAYTRACE. Filenames are currently configured to be no more than ten characters long.

The other menu files contain the actual information that is displayed on the console as a menu. As with MASTER.MEN, the other .MEN files are just text files. The information needed for the display of a menu is as follows. The first line has two items that are separated by a comma. These are the number of items in the menu, followed by the title of the menu. The number of items is an integer. All of the text information in these files is read in using an A40 format, and so may be no longer than 40 characters long. After this first line, the menu items are given. The text that will be displayed for each item is put in the file with one menu item's text per line. Again, these are to be no longer than 40 characters. The last line of the file contains the text for the prompt that will be displayed at the end of the menu. This information is accessible so that the menu text may be customized at will by the user with just an ordinary text editor.

A more complete understanding of how this all works will only be obtained by examining the source code of subroutine IOxMENU. In particular, this routine should be examined very closely if any extensions to RAYTRACE are to be made. Any additional information that must be entered ought to be entered by means of a menu so that the user interface of the program will remain consistent. Examination of the existing menu files will also be of assistance.

Appendix B:
Inventory of COMMON blocks

This section lists the COMMON blocks used by RAYTRACE, along with the declarations of the variables involved. An explanation is given for some of the variables. A complete explanation of all the variables involved would require describing significant portions of the algorithms used, which is beyond the scope of this User's Guide.

MAINDAT:

This block contains the contents of RAYDAT.DAT. As may be seen from the EQUIVALENCE statements, the array RVxIONPT holds the six sets of ionospheric parameters input from the data file.

```
INTEGER IVxPARAM(6), IVxLAUN(3)

REAL*8 RVxGRID(6), RVxIONPT(30,60,6), RVxLAUN(7), RVxOPTION(7
REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)

EQUIVALENCE (RVxION1,RVxIONPT),(RVxION2,RVxIONPT(1,1,2))
EQUIVALENCE (RVxION3,RVxIONPT(1,1,3))
EQUIVALENCE (RVxION4,RVxIONPT(1,1,4))
EQUIVALENCE (RVxION5,RVxIONPT(1,1,5))
EQUIVALENCE (RVxION6,RVxIONPT(1,1,6))

COMMON /MAINDAT/ RVxGRID, RVxIONPT, IVxPARAM, IVxLAUN,
* RVxLAUN, RVxOPTION
```

RESULTS:

The RESULTS common block contains the array which holds the data for every data point on a particular ray. For each ray that is traced, this data array is recorded. Refer to the discussion above concerning the FOR###.DAT file format for further detail on this array.

```
REAL*8 RVxBOUN(11,15)

COMMON /RESULTS/ RVxBOUN
```

LPARM:

```
      INTEGER IVxSSNUM, IVxTIME(5), IFxGRND
      REAL*8 RVxYEAR, RVxA100, RVxTIM
      COMMON /LPARM/ IVxSSNUM,IVxTIME,RVxYEAR,RVxA100,IFxGRND
     *,RVxTIM
```

LOSSES:

This block holds the various components of signal loss that are computed by RAYTRACE. RVxLOSA is the absorption loss. The code to calculate this is based upon empirical formulae which were appropriate to use at earlier stages of the development of RAYTRACE, but cannot be used for general paths. This code is currently commented out. RVxLOSX has suffered a similar fate. This was used to represent the so-called "excess loss" that some earlier ionospheric models use. This value is inappropriate, though, for use in examining specific raypaths in that it becomes possible for two relatively adjacent rays to have anomalously different losses. The remaining two values are actually used if signal intensity calculations are to be performed at all. RVxLOSR gives the reflection loss, and RVxLOSG gives the geometric loss. The geometric loss is the combination of the effects of distance from the source, and also focusing/defocusing effects.

```
      REAL*8 RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG
      COMMON /LOSSES/ RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG
```

PRAM:

The values in this block are, in order: pi, a conversion factor to go from degrees to radians, the radius of the earth, and the height considered to be the top of the ionosphere.

```
      REAL*8 RPxPI,RPxDTOR,RPxREARTH,RPxHTOP
      COMMON /PRAM/ RPxPI, RPxDTOR, RPxREARTH, RPxHTOP
```

SCPS1:

This block contains information on the number and location of the grid points. ICxNSCP is the number of grid points, and the other two are arrays containing the latitudes and longitudes, respectively, of the grid points.

```
INTEGER ICxNSCP

REAL*8 RVxLATSC(1800),RVxLONSC(1800)

COMMON /SCPS1/ ICxNSCP, RVxLATSC, RVxLONSC
```

SCPS1A:

These arrays contain the six ionospheric parameters for all the grid points, with all the layer peak plasma frequency squared values in one array and all the height (and semithickness) values in the second.

```
REAL*8 RVxFNSQ(1800,3), RVxH(1800,3)

COMMON /SCPS1A/ RVxFNSQ,RVxH
```

OTHER:

```
REAL*8 RVxLAT1, RVxLON1, RVxHBOT
REAL*8 RVxFSQU, RVxRPI, RVxHCUT

COMMON /OTHER/ RVxLAT1,RVxLON1,RVxHBOT,RVxFSQU,RVxRPI,RVxHCUT
```

MISC:

```
REAL*8 RVxSEZGEC(3,3), RVxX7, RVxY7, RVxZ7
REAL*8 RVxR1, RVxR2, RVxHMIN

COMMON /MISC/ RVxSEZGEC,RVxX7,RVxY7,RVxZ7,RVxR1,RVxR2,RVxHMIN
```

START:

This block contains the initial position information for the raypath increment.

```
REAL*8 RVxXI, RVxYI, RVxZI, RVxLATI, RVxLONI

COMMON /START/ RVxXI, RVxYI, RVxZI, RVxLATI, RVxLONI
```

END:

This block contains information on the end of the raypath increment.

```
REAL*8 RVxH5, RVxXF, RVxYF, RVxZF

COMMON /END/ RVxXF, RVxYF, RVxZF, RVxH5
```

IONO1:

This block stores some ionospheric parameters.

```
REAL*8 RVxALPHA, RVxBETA, RVxETA1, RVxETA2
REAL*8 RVxFNSB(3), RVxHB(3)

COMMON /IONO1/ RVxALPHA,RVxBETA,RVxETA1,RVxETA2,RVxFNSB,RVxHB
```

IONO2:

This block stores ionospheric interpolation results.

```
REAL*8 RVxB5(3), RVxB6(3), RVxA5(3), RVxA6(3)
REAL*8 RVxE5(3), RVxE6(3)

COMMON /IONO2/ RVxA5, RVxA6, RVxB5, RVxB6, RVxE5, RVxE6
```

IONO3:

This holds still other quantities used in the ionospheric calculations.

```
INTEGER IFxCASE
REAL*8 RVxV1, RVxV2, RVxXX, RVxHBND
COMMON /IONO3/ RVxV1,RVxV2,RVxXX,IFxCASE, RVxHBND
```

MORE:

```
INTEGER IVxSX, IVxSY, IVxSZ
REAL*8 RVxCX, RVxCY, RVxCZ
COMMON /MORE/ IVxSX, IVxSY, IVxSZ, RVxCX, RVxCY, RVxCZ
```

TEMP1:

```
REAL*8 RVxF40, RVxF65, RVxK1, RVxHL
COMMON /TEMP1/ RVxF40, RVxF65, RVxK1, RVxHL
```

GORP:

```
REAL*8 RVxDD6, RVxANGLIM
COMMON /GORP/ RVxDD6, RVxANGLIM
```

TEMP2:

These are indices use in accessing grid arrays for interpolation.

```
INTEGER IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
```

TEMP3:

```
REAL*8 RTLxD5, RTLxI3, RTLxI4, RTLxI5
COMMON /TEMP3/ RTLxI3, RTLxI4, RTLxI5, RTLxD5
```

VAR1:

```
REAL*8 RVCxXL, RVCxXU, RVCxA0, RVCxHU, RVCxB0
COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
```

VAR2:

```
REAL*8 RVCxHB1, RVCxH2, RVCxYS, RVCxSL1, RVCxSL2, RVCxH1L
COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS.RVCxSL1,RVCxSL2,RVCxH1L
```

VAR3:

```
REAL*8 RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
```

VAR4:

```
REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4, RVCxHT5
COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
```

RAID:

```
INTEGER IFCxSN
COMMON /RAID/ IFCxSN
```

TEMP4:

```
REAL*8 RTLxF3, RTLxF1, RTLxF2
COMMON /TEMP4/ RTLxF1, RTLxF2, RTLxF3
```

LOCAL:

```
REAL*8 RVLxELEV
COMMON /LOCAL/ RVLxELEV
```

Appendix C:
Error Messages

RAYTRACE will detect a few errors internally and generate error messages for these. These are in addition to those that may be generated as run-time errors as specified by the compiler being used, or may be operating system level errors. This section will discuss the error messages generated directly by RAYTRACE.

INVALID CHOICE.

This message indicates that a bad choice has been made from one of the menus. A bad choice is one that is out of the range of the possible selections.

HIT RETURN ONLY TO CONTINUE.

The routine that handles the 'Press RETURN to continue.' prompt will generate this message if anything other than an 'A', '.', or return is typed.

Conversion ERROR, Please RETRY Input

Any given prompt is expecting a particular type of response (REAL, INTEGER, etc.) and this message will be given if the value entered is not of the expected type. Note that if a REAL is expected, and an integer is entered, this will not occur, as a decimal point may be attached to the value.

RAY DOES NOT PROPAGATE!!!!

This message indicates that a ray has been given a starting point at an altitude inside the ionosphere and has parameters that are inconsistent with any propagation for the ray.

EXCEEDINGLY WIERD CASE: POINT #1 (& a few values are also displayed)

If this message ever appears, something has gone quite wrong in the routine which calculates the ionospheric parameters and gradients. Theoretically, this should never be seen. Copy down ALL the values given, along with all other parameters from the menus and the ionospheric specification (if possible). Contact the authors of RAYTRACE with this information.

EXCEEDINGLY WIERD CASE: POINT #2

This message is very similar in nature to the previous one. It, too, should never occur. If it does, however, follow the instructions given for the similar error above.

Appendix D:
Source Code Listing

APPEARS AFTER INDEX

References

[1] Reilly, M.H. and E.L. Strobel, "Efficient ray-tracing through a realistic ionosphere," to be published in Radio Sci., 1988.

[2] Thomason, J., G. Skaggs, and J. Lloyd, "A global ionospheric model," NRL Report 8321 (AD-000-323), August 1979.

Index

RAYTRACE USER'S GUIDE

APPENDIX D:

SOURCE CODE LISTING

Table of Contents

Naming Convention

Variables in the source code are generally named according to the following convention. In some cases, variables do not fall neatly into an appropriate category and the naming necessarily becomes a bit arbitrary. Cases that do not follow this convention at all are ones contained in code that has been obtained from other sources. Names are defined by:

(2 or 3-char prefix) (separator char) (mnemonic suffix)

(Type) (Purpose) (Context) x (Mnemonic)

Type:	I	INTEGER*4
	J	INTEGER*8
	F	REAL*4
	R	REAL*8
	C	COMPLEX*16
	K	CHARACTER
	L	LOGICAL
Purpose:	C	Constant
	V	Variable
	T	Temporary
	P	Parameter
	F	Flag
Context:	none	in MAIN program
	L	local to a subprogram
	S	passed as an argument
	C	passed in a COMMON block

EXAMPLES:	**RTLxH**	==	Real*8, temporary, local variable H
	IVxELLIM	==	Integer*4, variable ELLIM in MAIN

Line Numbering Convention

Line numbers are generally designated as follows:

```
        #    #    #    #    #
       ---  ---  ---------------
      ___|  |__       |_______________
designates   non-zero      for general use as in FOR-loops,
major        indicates     branching, and with FORMATs
subdivisions FORMAT stmt.
```

Flag Variables

IFxEXI: 1 indicates user's choice to quit the program, 0 otherwise

IFxNEW: 1 indicates a new problem is being done, 0 otherwise

IFxSTAY: 1 indicates that the user is continuing a problem, 0 otherwise

IFxBUN: # of extra ray in ray bundle for calculation of intensity

IFxGEN:
- 1 = no special condition exists
- 2 = ray increment has been adjusted so that a layer boundary is not overshot
- 3 = a z-reflection has occurred
- 4 = an x-reflection has occurred
- 5 = a y-reflection has occurred
- 6 = ray is traveling downward in free space propagation
- 7 = ray is traveling upward in free space propagation

IFxN4:	1 if the present location is not interior to a triangle of grid points for interpolation 2 otherwise
IFxGRND:	1 if reflection loss calculations are to use soil's conductivity 2 if water's conductivity is to be used
IFxSFL:	0 to activate signal intensity calculations 1 to deactivate them
IFLxOUT:	1 if the present location is outside the grid of points 0 otherwise

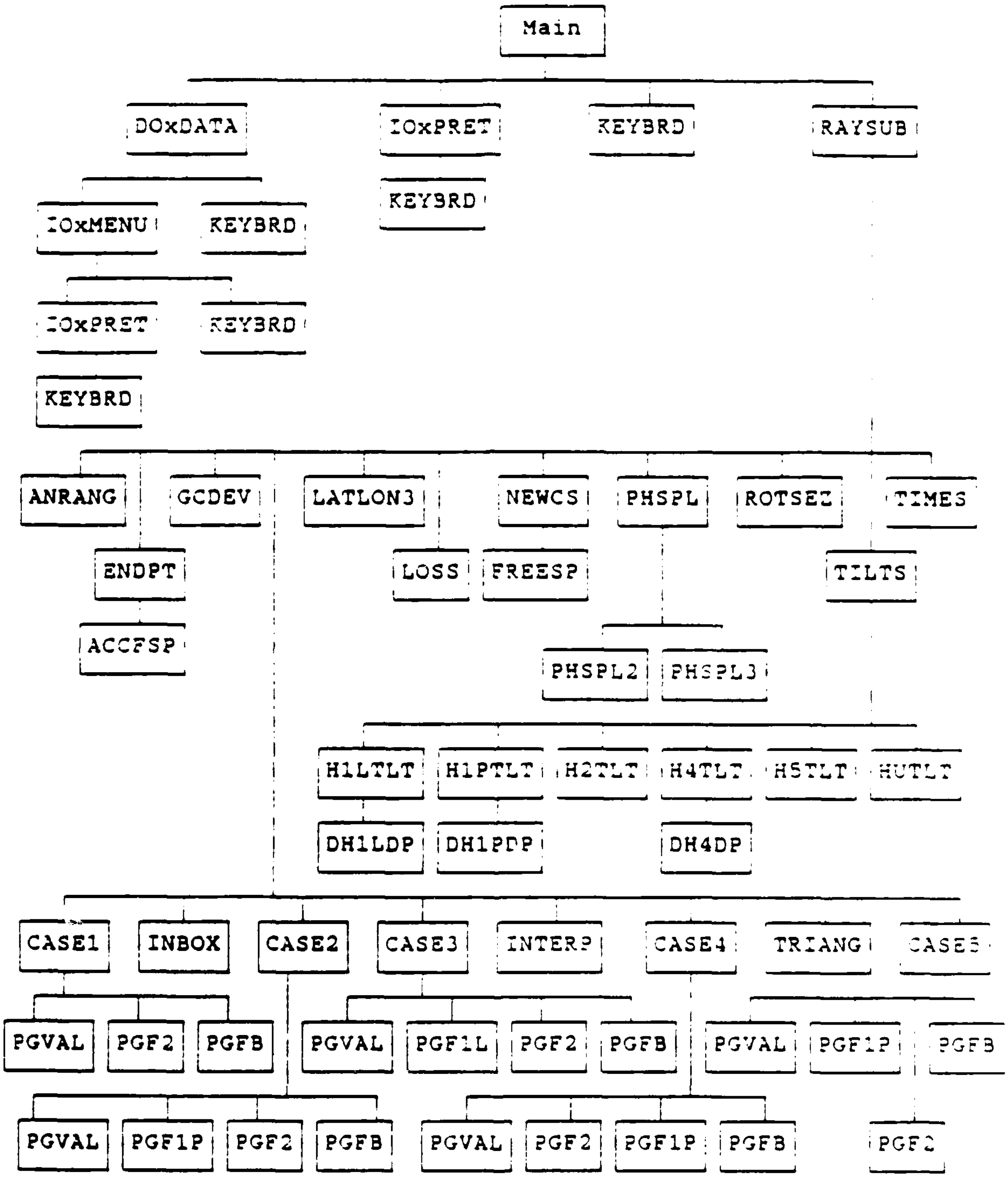
Notes

There will be discrepancies in the consecutive numbering of source code lines within a source code file, as the blank lines between subroutines have been skipped. Also, the consecutive numbering runs to the end of a particular source file, starting over for the next file.

The symbol table after each subroutine contains a column labeled "Class". This corresponds to the portion of the variable prefix denotes as "Context" in the naming convention above. The notable difference being that a Class of 'param' is the same as a prefix of 'S', meaning that the quantity was passed as an argument.

Also in the symbol table, the character 'x', which is used under the variable naming convention to connect the prefix with the mnemonic suffix, has been converted to an upper case letter, 'X'.

Hierarchy of Routines



Line# Source Line

```
      PROGRAM RAYTRACE
C
C ----------------------------------------------------------------
C
C   RAYTRACE --  3D RAYTRACING WITH ACCELERATED FREE SPACE
C                PROPAGATION CALCULATION
C
C ----------------------------------------------------------------
C
C   AUTHORS:        MICHAEL H. REILLY & ERIC L. STROBEL
C
C   DATE:           03/18/88
C
C   VERSION:        4.3
C
C ----------------------------------------------------------------
C
C   REVISED:       07/25/86 -- Initial revision. Translated
C                  from Tektronix Basic to VAX Fortran by
C                  Eric L. Strobel.
C
C                  07/30/86 -- V1.1.  Change over to use of
C                  REAL*8 precision in the calculations.
C
C                  08/12/86 -- V2.0.  Add the capability to
C                  perform propagation loss calculations.
C                  Also greatly massaged over the output.
C                  A number of minor and MAJOR bugs corrected.
C
C                  09/05/86 -- V2.3.  Add horizon focusing and
C                  handle low angle rays.  Record data for
C                  proper handling of skip focusing.  Record
C                  maximum height.  Calculate deviation from
C                  great circle path.  Minor bug fixes.
C
C                  09/17/86 -- V2.4.  Fix calculation of great
C                  circle path deviation.  Introduce cutoff in
C                  number of bounces.  Automagically obtain
C                  approximate bottom of the ionosphere, elim-
C                  inating the need for the prompt.  Also fixed
C                  rotation matrix used in calculating the new
C                  c-values.
C
C                  10/09/86 -- V3.0.  Lots of changes.  Read &
C                  write to files.  User-friendly means of
C                  getting information into the program.  The
C                  old Raytrace is now a subroutine.  Capability
C                  to do many rays at a time.  Able to handle
C                  large ionospheric specifications.  Able to
C                  change problem parameters without restarting
```

Line# Source Line

```
C                   the program.
C
C                   04/09/87 -- V3.1.  Corrected error in loss
C                   calculation.
C
C                   09/01/87 -- V4.0.  Major revision.  Now uses
C                   the RADAR-C ionosphere.  Able to launch rays
C                   inside and above the ionosphere.  Calcs. & rpts.
C                   the direction cosines now.  Uses new routine to
C                   obtain user info.  Absorption & excess loss calcs.
C                   commented out.  Geometric loss now done correctly.
C                   Phase path calculated. Data files now BINARY
C                   (vs. ASCII).  Upper bound at radius of GEO dist.
C                   Derivatives of range w.r.t. elev. are incorrect,
C                   but unfixed at this point.  WARNING: Cannot send
C                   rays arbitrarily close to vertical!!!  C-values
C                   are now properly normalized.
C
C                   09/30/87 -- V4.1.  Phase path length calculations
C                   added.  Tilt routines changed to conform to the
C                   requirements of using the RADAR-C model.
C
C                   01/15/88 -- V4.2.  Bug fixes and minor corrections
C                   made over a period of time.  Added feature: will
C                   now stop the ray at the designated range, not at
C                   the end of the next increment, as it previously
C                   did.  Also, some disused variables have been
C                   weeded out and some arrays which were larger than
C                   currently necessary have been redimensioned.  The
C                   code is now smaller and it runs faster.  The source
C                   code is now somewhat universal, in that only the
C                   slightest of changes is needed to convert from
C                   VAX to DOS or back, or between various DOS Fortrans.
C
C                   03/18/88 -- V4.3.  A number of corrections and
C                   improvements have been made.  Some elevation and
C                   azimuth options were being reset when they shouldn't
C                   have been (under some circumstances).  Angular range
C                   is now correctly calculated.  Before, it was based on
C                   the launch azimuth, which isn't correct if the ray
C                   bends.  Now, the actual azimuth between the two points
C                   is calculated and used.  Also, the actual angular
C                   range, and not the cosine of the angular range, is
C                   used for the range cutoff.
C
C --------------------------------------------------------
C
C                     Performs 3-D raytracing of radio propaga-
C         tion through the ionosphere.  Incorporates a spec-
C         ific model for true height profiles of electron
```

Line# Source Line

```
C        density.  Can utilizes data from ionospheric sounders.
C
C -----------------------------------------------------------------
C
        INTEGER ITxQ, IFxNEW, IFxSTAY, IFxEXI, IVxIND
        INTEGER IVxAZLIM, IVxELLIM, IVxPARAM(6), IVxLAUN(3)
        INTEGER IVxVAR, IVxTYP
C
        REAL*8 RVxGRID(6), RVxIONPT(30,60,6), RVxLAUN(7)
        REAL*8 RVxOPTION(7), RVxAZI, RVxELEV
        REAL*8 RVxBOUN(11,15), RVxVAR
        REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
        REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
C
        CHARACTER*10 KVxDFIL, KTxTMP, KVxOFIL
        CHARACTER*2 KVxIND
        CHARACTER*1 KTxANS, CH
        CHARACTER*40 KVxPMT
C
        COMMON /MAINDAT/ RVxGRID, RVxIONPT, IVxPARAM, IVxLAUN,
     * RVxLAUN, RVxOPTION
        COMMON /RESULTS/ RVxBOUN
C
C        Stuff the big array with pieces of more manageable size.
C          This is in order to compensate for the unfortunate DOS
C          restrictions on record sizes.
C
        EQUIVALENCE (RVxION1,RVxIONPT),(RVxION2,RVxIONPT(1,1,2))
        EQUIVALENCE (RVxION3,RVxIONPT(1,1,3))
        EQUIVALENCE (RVxION4,RVxIONPT(1,1,4))
        EQUIVALENCE (RVxION5,RVxIONPT(1,1,5))
        EQUIVALENCE (RVxION6,RVxIONPT(1,1,6))
C
C -----------------------------------------------------------------
C
C       Data setup section
C
C -----------------------------------------------------------------
C
        KVxPMT = ' '
        IVxTYP = 0
        ITxQ = 0
        IVxVAR = 0
        RVxVAR = 0.0
        KTxTMP = ' '
        IVxIND = 40
10000   IVxELLIM = 0
        IVxAZLIM = 0
        RVxAZI = 0.0D00
        RVxELEV = 0.0D00
```

Line# Source Line

```
C
C       IFxSTAY = 1 indicates that the user has responded with
C         'yes' to the query about performing another problem.
C
        IF (IFxSTAY.EQ.1) GO TO 20000
        WRITE (6,11001)
10005   KVxPMT = 'Is this a new (1) or old (0) problem? '
C
C       A new problem is one for which no RAYDAT.DAT or similar
C         file exists.  Such a file does exist for an old problem
C         and contains the user inputs from a previous run.
C
        CALL KEYBRD(KVxPMT,1,ITxQ,IFxNEW,RVxVAR,KTxTMP)
        IF (ITxQ.LT.1.OR.(.NOT.(IFxNEW.EQ.1.OR.IFxNEW.EQ.0))
     *  GO TO 10005
C
C ------------------------------------------------------------
C
C       Section to handle the set up of the problem's data
C         by reading from file and editing some of the
C         data.
C
C ------------------------------------------------------------
C
        IF (IFxNEW.EQ.1) THEN
C
C -------------------------------------------------------------
C       IFxNEW = 1    -->  new problem, so DOxDATA to get the user
C          options and ionospheric data, then write out to
C          a datafile.
C -------------------------------------------------------------
C
            WRITE (6,11004)
            KVxDFIL = 'RAYDAT.DAT'
            KVxPMT = 'information (10 char max):'
C
C       Ask the user where to store the problem's information.
C
            CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KVxDFIL)
            OPEN (30, FILE=KVxDFIL, FORM='UNFORMATTED',STATUS='NEW'
            CALL DOxDATA(IFxSTAY, IFxNEW, IFxEXI)
            REWIND (30)
            WRITE (30) RVxGRID
            WRITE (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
            WRITE (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
            WRITE (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
            WRITE (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
            WRITE (30) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
            WRITE (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
            WRITE (30) IVxPARAM
```

Line# Source Line

```
              WRITE (30) IVxLAUN
              WRITE (30) RVxLAUN
              WRITE (30) RVxOPTION
              CLOSE(30)
C
C         IFxEXI = 1 indicates that the user has chosen to quit the
C           program.
C
              IF (IFxEXI.EQ.1) GO TO 20700
          ELSE IF (IFxNEW.EQ.0) THEN
C
C ------------------------------------------------------------
C          IFxNEW = 0  -->  old problem, so read from the datafile,
C                 then DOxDATA to check for any changes the user
C                 wants to make to the options.
C ------------------------------------------------------------
C
              WRITE (6,11004)
              KVxDFIL = 'RAYDAT.DAT'
              KVxPMT = 'information (10 char max):'
C
C         Ask the user where to store the problem's information.
C
              CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KVxDFIL)
              OPEN (30, FILE=KVxDFIL, FORM='UNFORMATTED', STATUS='OLD')
              REWIND (30)
              READ (30) RVxGRID
              READ (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
              READ (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
              READ (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
              READ (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
              READ (30) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
              READ (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
              READ (30) IVxPARAM
              READ (30) IVxLAUN
              READ (30) RVxLAUN
              READ (30) RVxOPTION
              CALL DOxDATA(IFxSTAY, IFxNEW, IFxEXI)
              IF (IFxEXI.EQ.1) THEN
C
C         The user has chosen to quit, so close the file and quit.
C
                  CLOSE (30)
                  GO TO 20700
              ENDIF
10010         KVxPMT = 'Overwrite existing file (Y/N)? '
C
C -------------------------------------------------------------
C           Give the user the choice of replacing the old problem's
C                data with the (possibly new) current data, or
```

```
Line#  Source Line

C               making a second (backup) datafile.
C ----------------------------------------------------------------
C
                  CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KTxANS)
                  IF (KTxANS.EQ.'Y') THEN
                      GO TO 10015
                  ELSE IF (KTxANS.EQ.'N') THEN
                      CLOSE(30)
                      KTxANS = ' '
                      KVxPMT = 'Backup the datafile (Y/N)? '
C
C             If the user has decided not to overwrite the old data, ask
C               if a separate copy is to be made...
C
                      CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KTxANS)
                      IF (KTxANS.EQ.'N') GO TO 20000
                      KVxPMT = 'New filename (10 char max)? '
C
C             ... and if so, ask what the new file's name will be.
C
                      CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KVxDFIL)
                      OPEN (30,FILE=KVxDFIL,FORM='UNFORMATTED',STATUS='NEW'
                  ELSE
C
C             The user has given an unexpected response, so go back and
C               do it over again.
C
                      GO TO 10010
                  ENDIF
      10015       REWIND (30)
                  WRITE (30) RVxGRID
                  WRITE (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
                  WRITE (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
                  WRITE (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
                  WRITE (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
                  WRITE (30) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
                  WRITE (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
                  WRITE (30) IVxPARAM
                  WRITE (30) IVxLAUN
                  WRITE (30) RVxLAUN
                  WRITE (30) RVxOPTION
                  CLOSE(30)
              ELSE
C
C              Else this is an unexpected answer, so re-prompt.
C
                  GO TO 10005
              ENDIF
      11001   FORMAT (//////////////////////////////////)
      11004   FORMAT ('0Enter filename for storage of ionospheric'
```

```
Line#  Source Line

C
C ------------------------------------------------------------------
C
C          End data set up section.
C
C ------------------------------------------------------------------
C
20000    IF (IFxSTAY.EQ.1) THEN
C
C ------------------------------------------------------------------
C          If we've been running and decide to do another problem,
C              (STAY in the program) re-cycle starts here.  DOxDATA
C              to get the (possibly new) user values for the next
C              problem & give the user the opportunity to update
C              the datafile.
C ------------------------------------------------------------------
C
              CALL DOxDATA(IFxSTAY, IFxNEW, IFxEXI)
C
C        IFxEXI = 1 indicates that the user has decided to quit.
C
              IF (IFxEXI.EQ.1) GO TO 20700
              KTxANS = ' '
              KVxPMT = 'Update the datafile (Y/N)? '
              CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KTxANS)
              IF (KTxANS.EQ.'Y') THEN
                  OPEN (30, FILE = KVxDFIL, FORM='UNFORMATTED')
                  REWIND (30)
                  WRITE (30) RVxGRID
                  WRITE (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                  WRITE (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                  WRITE (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                  WRITE (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                  WRITE (30) ((RVxION5(I,J),J=1 RVxGRID(6)),I=1
     #,RVxGRID(5))
                  WRITE (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                  WRITE (30) IVxPARAM
                  WRITE (30) IVxLAUN
                  WRITE (30) RVxLAUN
                  WRITE (30) RVxOPTION
                  CLOSE(30)
              ENDIF
              KTxANS = ' '
          ENDIF
C
```

Line# Source Line

```
C -----------------------------------------------------------------
C
C       Begin actually firing out some rays.  Results are stored
C         in consecutively numbered files, FORnnn.DAT, starting
C         with  nnn = 040.  Beware that if the program is run on
C         DOS or similar systems, starting another run of the
C         program may cause data loss, since the new result files
C         will simply write over the old ones.
C
C -----------------------------------------------------------------
C
C        First, select the appropriate limits on the DO loops.
C
        IF (IDNINT(RVxOPTION(1)).EQ.0) THEN
            IVxELLIM = 1
            IVxAZLIM = 1
        ELSE IF (IDNINT(RVxOPTION(1)).EQ.1) THEN
            IVxAZLIM = 1
            IVxELLIM = 1+IDNINT((RVxOPTION(7)-RVxOPTION(3))/
     #RVxOPTION(5))
        ELSE
            IVxELLIM = 1+IDNINT((RVxOPTION(7)-RVxOPTION(3))/
     #RVxOPTION(5))
            IVxAZLIM = 1+IDNINT((RVxOPTION(6)-RVxOPTION(2))/
     #RVxOPTION(4))
        ENDIF
        PRINT *,' '
        PRINT *,' BEGINNING RAY LOOPS.'
C
C       Generate the name of the result file by putting the number
C         part into an 'internal device' (a string).  This string
C         is then concatenated with the rest of the file's name.
C
        WRITE (KVxIND,'(I2)') IVxIND
        KVxOFIL = 'FOR0'//KVxIND//'.DAT'
        OPEN (IVxIND,FILE = KVxOFIL)
        WRITE (IVxIND, 22000) IVxLAUN(1)+1, IVxAZLIM, IVxELLIM
C
C       These first numbers are written to the result file so that
C         an analysis program can determine how many points of
C         information there are.
C
        DO 20200 I = 1, IVxAZLIM
            DO 20100 J = 1, IVxELLIM
                RVxAZI = RVxOPTION(2) + (I-1)*RVxOPTION(4)
                IF (RVxAZI.GE.360.0D00) RVxAZI = RVxAZI - 360.0
                RVxELEV = RVxOPTION(3) + (J-1)*RVxOPTION(5)
                WRITE(IVxIND, 21003) RVxAZI, RVxELEV
C
C       Preface the results for a ray with the azimuth and elevation.
```

```
Line#  Source Line

C
C ----------------------------------------------------------------
C
C       RAYSUB is the routine that actually traces the rays.
C
                CALL RAYSUB(RVxAZI, RVxELEV)
C
C       Now, write the results of the ray.  The elements of the
C         result array are discussed in the comment block near
C         the beginning of RAYSUB.
C
                WRITE(IVxIND, 21003) ((RVxBOUN(L,K),K=1,15), L=1,
     # IVxLAUN(1)+1)
                DO 20099 K = 1, 15
                    DO 20098 L = 1, IVxLAUN(1)+1
C
C        Clear the result array for the next ray.
C
                        RVxBOUN(L,K) = 0.0
20098               CONTINUE
20099           CONTINUE
20100       CONTINUE
20200   CONTINUE
        CLOSE(IVxIND)
        PRINT *,' '
        PRINT *,' RAY LOOPS DONE.'
        PRINT *,' '
        WRITE (6,21004) KVxOFIL
C
C        Increment the unit number that the results will be
C            written to.
C
        IVxIND = IVxIND + 1
C
C ----------------------------------------------------------------
C
C        End section that shoots rays.
C
C ----------------------------------------------------------------
C
20500   KVxPMT = 'Do another problem (Y/N)?'
C
C        Give the user a chance to quit.
C
        KTxANS = ' '
        CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KTxANS)
        IF (KTxANS.EQ.'Y') THEN
            IFxSTAY = 1
            GO TO 10000
        ELSE IF (ITxQ.LT.1.OR.KTxANS.NE.'N') THEN
```

Line# Source Line

```
                        GO TO 20500
                    ENDIF
                    KTxANS = ' '
        20700       CALL IOxPRET(CH)
        C
        C            Now, give the user the chance to abort the quit.
        C
                    IF (CH.EQ.'A') GO TO 20500
        C
        C            Print the closing message.
        C
                    WRITE (6,21007)
                    WRITE (6,21008)
                    WRITE (6,21009)
        21002       FORMAT (A1)
        21003       FORMAT (G24.14)
        21004       FORMAT (///,1X,'Results written to file ',A10,//)
        21007       FORMAT (1X,'Session done.  For safety''s sake, copy the data'
        21008       FORMAT (1X,'and result files into a separate directory to')
        21009       FORMAT (1X,'prevent accidental overwriting.')
        22000       FORMAT (I4)
                    END
```

main Local Symbols

Name	Class	Type	Size
ITXQ.	local	INTEGER*4	4
I	local	INTEGER*4	4
J	local	INTEGER*4	4
K	local	INTEGER*4	4
CH.	local	CHAR*1	1
L	local	INTEGER*4	4
IFXEXI.	local	INTEGER*4	4
IFXNEW.	local	INTEGER*4	4
IVXIND.	local	INTEGER*4	4
KVXIND.	local	CHAR*2	2
KVXDFIL	local	CHAR*10	10
KTXANS.	local	CHAR*1	1
IVXVAR.	local	INTEGER*4	4
KVXOFIL	local	CHAR*10	10
RVXAZI.	local	REAL*8	8
KTXTMP.	local	CHAR*10	10
IFXSTAY	local	INTEGER*4	4
RVXVAR.	local	REAL*8	8
KVXPMT.	local	CHAR*40	40
IVXELLIM.	local	INTEGER*4	4
RVXELEV	local	REAL*8	8
IVXTYP.	local	INTEGER*4	4

main Local Symbols

Name	Class	Type	Size
IVXAZLIM.	local	INTEGER*4	4
IVXPARAM.	MAINDAT	INTEGER*4	24
IVXLAUN	MAINDAT	INTEGER*4	12
RVXGRID	MAINDAT	REAL*8	48
RVXIONPT.	MAINDAT	REAL*8	86400
RVXLAUN	MAINDAT	REAL*8	56
RVXOPTION	MAINDAT	REAL*8	56
RVXBOUN	RESULTS	REAL*8	1320
RVXION1	MAINDAT	REAL*8	14400
RVXION2	MAINDAT	REAL*8	14400
RVXION3	MAINDAT	REAL*8	14400
RVXION4	MAINDAT	REAL*8	14400
RVXION5	MAINDAT	REAL*8	14400
RVXION6	MAINDAT	REAL*8	14400

Line# Source Line

```
      SUBROUTINE DOxDATA(IFSxSTA, IFSxNEW, IFSxEXI)
C
C-----------------------------------------------------------------
C
C   DOxDATA -- SUBROUTINE TO MAINTAIN AND ACCESS THE DATA FOR
C         CONVENIENT USE OF THE WHOLE RAYTRACING PROGRAM
C
C         CALLED BY: MAIN
C
C         CALLS: IOxPRET, KEYBRD
C
C-----------------------------------------------------------------
C
C         AUTHOR:        ERIC L. STROBEL
C
C         DATE:          01/15/88
C
C         VERSION:       2.1
C
C-----------------------------------------------------------------
C
C         REVISED:          10/09/86 -- V1.0.  Initial revision.
C
C                           09/01/87 -- V2.0.  Uses the new KEYBRD
C                           routine and unformatted reads and writes.
C
C                           01/15/88 -- V2.1.  Contains necessary
C                           concessions to the limited environment that
C                           DOS provides.
C
C-----------------------------------------------------------------
C
C       USES:   IFSxSTAY         A FLAG THAT INDICATES THAT THE
C                                 USER HAS REMAINED IN THE PROBLEM
C               IFSxNEW          A FLAG THAT INDICATES THAT THIS
C                                 IS A NEW PROBLEM
C
C        TO DECIDE HOW TO PRESENT AN INTERFACE TO THE USER, IN
C                ORDER TO MAINTAIN THE DATABASE FILES
C
C       RETURNS:         IFSxEXI          A FLAG THAT INDICATES THAT
C                                          THE USER HAS CHOSEN TO EXIT
C
C-----------------------------------------------------------------
C
      INTEGER IFSxSTA, IFSxNEW, IFSxEXI, ITLxQ
      INTEGER IVxPARAM(6), ITLxC1, ITLxC2, ITLxC3
      INTEGER ITLxTMP, IVxLAUN(3), IVLxTYP,IVLxVAR
C
      REAL*8 RVLxLAT, RVLxLON, RVxGRID(6), RVxIONPT(30,60,5
```

```
Line#  Source Line

            REAL*8 RTLx1, RTLx2, RVxLAUN(7), RVxOPTION(7), RTxA
            REAL*8 RVLxVAR, RPxDTOR
            REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
            REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
      C
            CHARACTER*10 KVLxFIL, KTLxTMP
            CHARACTER*40 KVLxPMT
            CHARACTER*1 KVLxANS
            CHARACTER*7 KCLxLBL(6)
      C
            COMMON /MAINDAT/ RVxGRID,RVxIONPT,IVxPARAM,IVxLAUN,
          # RVxLAUN,RVxOPTION
      C
      C       Stuff the big array with pieces of more manageable size.
      C         This is in order to compensate for the unfortunate DOS
      C         restrictions on record sizes.
      C
            EQUIVALENCE (RVxION1,RVxIONPT), (RVxION2,RVxIONPT(1,1,2))
            EQUIVALENCE (RVxION3,RVxIONPT(1,1,3))
            EQUIVALENCE (RVxION4,RVxIONPT(1,1,4))
            EQUIVALENCE (RVxION5,RVxIONPT(1,1,5))
            EQUIVALENCE (RVxION6,RVxIONPT(1,1,6))
      C
            DATA KCLxLBL /' foE**2','   hmF1','foF1**2','   YmF2',
          #'   hmF2','foF2**2'/
      C
      C -----------------------------------------------------------------
      C
            KVLxPMT = ' '
            IVLxTYP = 0
            IVLxVAR = 0
            RVLxVAR = 0.0
            KTLxTMP = ' '
            RPxDTOR = 0.0174532925D00
      C
      C -----------------------------------------------------------------
      C       If this is the first time thru on a new problem, read
      C          in either an ionospheric grid produced by another
      C          program (RADAR-C right now) or a grid that is to be
      C          entered by hand.
      C -----------------------------------------------------------------
      C
      10000 IF (IFSxNEW.EQ.1.AND.IFSxSTA.EQ.0) THEN
               WRITE(6,11001)
               WRITE(6,11002)
               KVLxFIL = 'GRID.DAT'
               KVLxPMT = ' (type NONE if none exists) '
               CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR,KVLxFIL)
               IF (KVLxFIL.EQ.'NONE') THEN
      C
```

Line# Source Line

```
C         No external grid file exists, so build it by hand.
C
                OPEN(20, FILE='GRID.DAT',FORM='UNFORMATTED'
     #, STATUS='NEW')
                REWIND (20)
10020           WRITE (6,'(1X,///////////////////////////////////////////)')
C
C         RVxGRID gives info needed to obtain the lat's & lon's
C               of the grid.
C
                KVLxPMT = 'Input lat grid spacing (deg): '
                RTLx1 = RVxGRID(1)
                RTLx2 = RVxGRID(2)
C
C        Prompt for the latitude spacing of the lat-lon grid.
C
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                IF (ITLxQ.GE.1) RVxGRID(1) = RTLx1
                KVLxPMT = 'Input lon grid spacing (deg): '
C
C        Prompt for the longitude spacing of the lat-lon grid.
C
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
                IF (ITLxQ.GE.1) RVxGRID(2) = RTLx2
C
                WRITE(6,11007)
                KVLxPMT = '                    LATITUDE :'
                RTLx1 = RVxGRID(3)
C
C        Prompt for the grid's starting latitude.
C
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                IF (ITLxQ.GE.1) RVxGRID(3) = RTLx1
C
                KVLxPMT = 'LONGITUDE (east = positive):'
                RTLx2 = RVxGRID(4)
C
C        Prompt for the grid's starting longitude.
C
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
                IF (ITLxQ.GE.1) RVxGRID(4) = RTLx2
C
                KVLxPMT = 'Input # of grid points in lat.: '
                RTLx1 = RVxGRID(5)
C
C        Prompt for the number of grid point in the latitude direction.
C
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                IF (ITLxQ.GE.1) RVxGRID(5) = RTLx1
C
```

```
Line#  Source Line

                 KVLxPMT = '                        ...  in lon.: '
                 RTLx1 = RVxGRID(6)
 C
 C       Prompt for the number of grid point in the longitude direction
 C
                 CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                 IF (ITLxQ.GE.1) RVxGRID(6) = RTLx1
 C
 C       Verify the grid inputs and give the user a chance to re-enter
 C         them to fix errors.
 C
                 KVLxANS = 'Y'
                 KVLxPMT = 'Grid setup OK (Y/N) ? '
                 CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR,KVLxANS)
                 IF (KVLxANS.EQ.'N'.OR.KVLxANS.EQ.'n') GO TO 10020
 C
 C       Write the results out into a GRID file.
 C
                 WRITE(20) RVxGRID
                 WRITE (6,'(1X,/////////////////////////////////////)')
 C
 C       DO-loops for the entry of ionospheric parameters at each
 C         grid point.
 C
                 DO 10200 I=1, RVxGRID(5)
                   RVLxLAT = RVxGRID(3) + (I-1)*RVxGRID(1)
                   DO 10100 J=1, RVxGRID(6
                     RVLxLON = RVxGRID(4) + (J-1)*RVxGRID(2)
                     WRITE (6,12002) RVLxLAT, RVLxLON
 10049               DO 10050 K=1,6
 C
 C       The elements of KCLxLBL may be found in the DATA statement
 C         at the beginning of this routine.
 C
                       KVLxPMT = 'Input '//KCLxLBL(K)//' :'
                       RTLx1 = RVxIONPT(I,J,K)
                       CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1
      #,KTLxTMP)
                       IF (ITLxQ.GE.1) RVxIONPT(I,J,K) = RTLx1
 10050               CONTINUE
                     KVLxANS = 'Y'
                     KVLxPMT = 'Profile inputs OK (Y/N)? '
 C
 C       For each grid point, verify correct entry of values and give
 C         the user a chance to fix errors.
 C
                     CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR
      #,KVLxANS)
                     IF (KVLxANS.EQ.'N'.OR.KVLxANS.EQ.'n') GO TO 10049
                     WRITE (6,'(1X,////////////////////////////////////)'
```

Line# Source Line

```
10100             CONTINUE
10200            CONTINUE
C
C         Write the grid file in the same manageable pieces that
C              everything else expects it to be in.
C
                 WRITE (20) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                 WRITE (20) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                 WRITE (20) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                 WRITE (20) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                 WRITE (20) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
                 WRITE (20) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1
     #,RVxGRID(5))
C
C       Do a series of prompts for the sunspot number and time values.
C
10201            KVLxPMT = 'Input (integer) sunspot number: '
                 ITLxTMP = IVxPARAM(1)
                 CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP)
                 IF (ITLxQ.GE.1) IVxPARAM(1) = ITLxTMP
C
                 KVLxPMT = 'Input year: '
                 ITLxTMP = IVxPARAM(2)
                 CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
                 IF (ITLxQ.GE.1) IVxPARAM(2) = ITLxTMP
C
                 KVLxPMT = '...  month: '
                 ITLxTMP = IVxPARAM(3)
                 CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
                 IF (ITLxQ.GE.1) IVxPARAM(3) = ITLxTMP
C
                 KVLxPMT = '...    day: '
                 ITLxTMP = IVxPARAM(4)
                 CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
                 IF (ITLxQ.GE.1) IVxPARAM(4) = ITLxTMP
C
                 KVLxPMT = 'Input UT time (hr) : '
                 ITLxTMP = IVxPARAM(5)
                 CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
                 IF (ITLxQ.GE.1) IVxPARAM(5) = ITLxTMP
C
                 KVLxPMT = '...   UT time (min): '
                 ITLxTMP = IVxPARAM(6)
                 CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
                 IF (ITLxQ.GE.1) IVxPARAM(6) = ITLxTMP
```

```
Line#   Source Line

C
                WRITE (20) IVxPARAM
                KVLxPMT = 'S.S. # and times OK (Y/N)? '
                KVLxANS = 'Y'
C
C       Again, give the user a chance to verify the entries and to
C         correct errors.
C
                CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR,KVLxANS)
                IF (KVLxANS.EQ.'N'.OR.KVLxANS.EQ.'n') GO TO 10201
C
C       Close the GRID file and go on to get the rest of the user's
C         input.
C
                CLOSE(20)
                GO TO 20000
            ENDIF
C
C         Read the externally constructed grid file.
C
            OPEN(20, FILE=KVLxFIL,FORM='UNFORMATTED')
            REWIND (20)
            READ(20) RVxGRID
            READ(20) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
            READ(20) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
            READ(20) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
            READ(20) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
            READ(20) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
            READ(20) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
            READ(20) IVxPARAM
            CLOSE(20)
        ENDIF
11001   FORMAT(/////////////////////////////////////////////////)
11002   FORMAT(1X,'The ionospheric grid file name is?')
11007   FORMAT(1X,'Input starting latitude and longitude in degrees.')
12002   FORMAT(/,1X,'GRID PT. - Lat: ',G15.6,'  - Lon: ',G15.6)
C
C ---------------------------------------------------------
C
C        Done reading in data, now build/alter database of
C          options, by processing menu choices.
C
C ---------------------------------------------------------
C
C        It should be noted that by simply adding to the list
C          of lines contained in the computed GO TOs, additional
C          items may be attached to the existing menus, or additional
C          menus may be added.  If the menu system is extended
C          in this way, the routine IOxMENU is of sufficient
C          generality to handle this.  The appropriate menu
```

```
Line#  Source Line
      C          files must be changed/added and care should be
      C          taken to make sure that any one menu list doesn't
      C          become so large that part of it scrolls up off the
      C          screen before the prompt comes up.  The menu file
      C          scheme is discussed in a comment block in the
      C          routine IOxMENU.
      C
      C ----------------------------------------------------------
      C
      C ----------------------------------------------------------
      C       MENU 1 -- Main menu:  4 choices
      C ----------------------------------------------------------
      C
      20000   CALL IOxMENU(1,ITLxC1)
              GO TO (20010, 20200, 20400, 20600), ITLxC1
      C
      C ----------------------------------------------------------
      C       MENU 2 -- Launch parameters menu:  9 choices
      C ----------------------------------------------------------
      C
      20010   CALL IOxMENU(2, ITLxC2)
              GO TO (20020,20030,20040,20050,20060,20070,20080
             #,20090,20000), ITLxC2
      20020   IF (IVxLAUN(1).EQ.0) IVxLAUN(1)=1
              KVLxPMT = 'Bounce limit '
              ITLxTMP = IVxLAUN(1)
      C
      C       Prompt for the limit in the number of times the ray
      C         will be allowed to return to earth, if this is in
      C         fact an applicable concept.  The default value is
      C         1.
      C
              CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP)
              IF (ITLxQ.GE.1) IVxLAUN(1) = ITLxTMP
              GO TO 20010
      20030   KVLxPMT = 'Do signal intensity (no = 1)? '
              ITLxTMP = IVxLAUN(2)
      C
      C       Prompt to find out whether or not the user wants signal
      C         intensity to be calculated.
      C
              CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP)
              IF (ITLxQ.GE.1) IVxLAUN(2) = ITLxTMP
              GO TO 20010
      20040   KVLxPMT = 'Reflect from ground(1) or water(0)? '
              ITLxTMP = IVxLAUN(3)
      C
      C       Prompt the user to choose a generic conductivity for the
      C         reflection loss calculation at an earth bounce point.
      C         Someday this should be replaced with a global conductivity
```

```
Line#  Source Line

      C            map.
      C
                CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP)
                IF (ITLxQ.GE.1) IVxLAUN(3) = ITLxTMP
                GO TO 20010
      20050     KVLxPMT = 'Enter launch pt. latitude '
      C
      C         Do a pair of prompts for the latitude, longitude location
      C           of the starting point of the ray.
      C
                RTLx1 = RVxLAUN(1)
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                IF (ITLxQ.GE.1) RVxLAUN(1) = RTLx1
                KVLxPMT = 'Enter launch pt. longitude '
                RTLx2 = RVxLAUN(2)
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
                IF (ITLxQ.GE.1) RVxLAUN(2) = RTLx2
                GO TO 20010
      20060     KVLxPMT = 'Enter starting ht. (km) '
                RTLx1 = RVxLAUN(6)
      C
      C         Prompt for the altitude of the starting point for the ray.
      C
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                IF (ITLxQ.GE.1) RVxLAUN(6) = RTLx1
                GO TO 20010
      20070     IF (RVxLAUN(3).EQ.0.0) RVxLAUN(3) = 3000.0D00
                IF (RVxLAUN(7).EQ.0.0) RVxLAUN(7) = 36000.0D00
      C
      C         A pair of prompts to establish the range and altitude
      C           cutoff values necessary to prevent the problem from
      C           running on forever.  The range is the ground range
      C           along the ground track of the ray, irregardless of the
      C           altitude and actual path of the ray.  The default
      C           value for the ground range is 3000 km, and the default
      C           for the altitude limit is 36000 km, or approximately
      C           geostationary altitude.
      C
                KVLxPMT = 'Enter range limit (km) '
                RTLx1 = RVxLAUN(3)
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                IF (ITLxQ.GE.1) RVxLAUN(3) = RTLx1
                KVLxPMT = 'Enter altitude limit (km) '
                RTLx2 = RVxLAUN(7)
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
                IF (ITLxQ.GE.1) RVxLAUN(7) = RTLx2
                GO TO 20010
      20080     IF (RVxLAUN(4).EQ.0.0) RVxLAUN(4) = 4.0D00
                KVLxPMT = 'Enter raypath increment (km) '
                RTLx1 = RVxLAUN(4)
```

Line# Source Line

```
C
C        Prompt for the size of the nominal raypath increment.
C          The default value of 4 km represents a compromise
C          between execution time (longer increments = faster
C          execution) and accuracy (shorter increments = more
C          accurate).
C
         CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
         IF (ITLxQ.GE.1) RVxLAUN(4) = RTLx1
         GO TO 20010
20090    KVLxPMT = 'Enter wave frequency (MHz) '
         IF (RVxLAUN(5).EQ.0.0) RVxLAUN(5) = 5.0D00
         RTLx1 = RVxLAUN(5)
C
C        Prompt for the frequency of the HF transmission that
C          the ray is supposed to represent.  The default value
C          of 5 MHz is somewhat arbitrary.  Since the program
C          performs frequent divisions by wave frequency squared,
C          some sort of non-zero default value is needed just to
C          prevent 'unexplained' crashes of the program for new
C          problems.
C
         CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
         IF (ITLxQ.GE.1) RVxLAUN(5) = RTLx1
         GO TO 20010
C
C ---------------------------------------------------------------
C        MENU 3 -- Elevation parameters menu;  8 choices.
C ---------------------------------------------------------------
C
20200    CALL IOxMENU(3,ITLxC3)
         GO TO (20210,20220,20230,20240,20250,20260,20270,20000),ITLxC3
20210    KVLxPMT = 'Enter dimension of problem '
         RTLx1 = RVxOPTION(1)
C
C        The dimension of the problem goes as follows (the reason for
C          the term dimension should become obvious):
C
C          0 = a single ray (a single point on the sky)
C          1 = a vertical fan of rays at a single azimuth
C              (a line of points on the sky)
C          2 = a set of fans of rays (an array of points on
C              the sky)
C
         CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
         IF (ITLxQ.LT.1) THEN
             GO TO 20200
         ELSE IF (IDNINT(RTLx1).GT.2.OR.IDNINT(RTLx1).LT.0) THEN
             GO TO 20210
         ELSE
```

```
Line#  Source Line

                   RVxOPTION(1) = RTLx1
                   GO TO 20200
               ENDIF
       20220   KVLxPMT = 'Enter starting azimuth (deg) '
               RTLx1 = RVxOPTION(2)
       C
       C       Prompt for the initial azimuth for the problem.
       C
               CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
               IF (ITLxQ.GE.1) RVxOPTION(2) = RTLx1
               GO TO 20200
       20230   WRITE (6,22009)
               KVLxPMT = 'for values >90 deg from zenith. '
               RTLx1 = RVxOPTION(3)
       C
       C       Prompt for the initial elevation for the problem.
       C         For a starting point at greater than zero altitude,
       C         elevation angles may be greater than 90 degrees
       C         from the zenith.  Such angles are designated by
       C         negative values.
       C
               CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
               IF (ITLxQ.GE.1) RVxOPTION(3) = RTLx1
               GO TO 20200
       20240   IF (IDNINT(RVxOPTION(1)).EQ.0) THEN
                   IF (RVxOPTION(5).EQ.0.0) RVxOPTION(5) = 1.0D00
                   GO TO 20200
               ENDIF
               KVLxPMT = 'Enter elev. resolution (deg) '
               RTLx1 = RVxOPTION(5)
       C
       C       Prompt for the spacing of rays in the elevation
       C         direction.  This prompt is only displayed if it
       C         is appropriate, namely if the dimension of the
       C         problem is 1 or 2.  Otherwise, a default value
       C         of 1 is set and the user is returned directly to
       C         the elevation parameters menu.
       C
               CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
               IF (ITLxQ.GE.1) RVxOPTION(5) = RTLx1
               GO TO 20200
       20250   IF (IDNINT(RVxOPTION(1)).LT.2) THEN
                   IF (RVxOPTION(4).EQ.0.0) RVxOPTION(4) = 1.0D00
                   GO TO 20200
               ENDIF
               KVLxPMT = 'Enter azimuth resolution (deg) '
               RTLx1 = RVxOPTION(4)
       C
       C       Prompt for the azimuth spacing of the rays.  This
       C         prompt is only displayed if it is appropriate.
```

Line#   Source Line

```
      C          namely if the dimension of the problem is 2.
      C          Otherwise, a default value of 1 is set and the user
      C          is returned directly to the elevation parameters
      C          menu.
      C
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                IF (ITLxQ.GE.1) RVxOPTION(4) = RTLx1
                GO TO 20200
      20260     IF (IDNINT(RVxOPTION(1)).EQ.0) THEN
                    IF (RVxOPTION(7).EQ.0.0) RVxOPTION(7) = RVxOPTION(3)
                    GO TO 20200
                ENDIF
                KVLxPMT = 'Enter elev. limit (deg) '
                RTLx1 = RVxOPTION(7)
      C
      C         Prompt to establish the other end of the range of
      C           elevation values to be stepped through, the range
      C           beginning with the starting elevation from above.
      C           This prompt is only displayed if appropriate, namely
      C           if there are to be multiple rays.  Otherwise, the
      C           limit is set to be equal to the starting elevation and
      C           the user is returned directly to the elevation
      C           parameters menu.
      C
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                IF (ITLxQ.GE.1) RVxOPTION(7) = RTLx1
                GO TO 20200
      20270     IF (IDNINT(RVxOPTION(1)).LT.2) THEN
                    IF (RVxOPTION(6).EQ.0.0) RVxOPTION(6) = RVxOPTION(2)
                    GO TO 20200
                ENDIF
                KVLxPMT = 'Enter azimuthal limit (deg) '
                RTLx1 = RVxOPTION(6)
      C
      C         Prompt for the other end of the range of azimuth values
      C           which will be stepped through, beginning with the
      C           starting azimuth value.  This prompt is only displayed
      C           when appropriate, i.e. when there the dimension of the
      C           problem is 2.  Otherwise, the limit is set to be equal
      C           to the starting azimuth and the user is returned
      C           directly to the elevation parameters menu.
      C
                CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
                IF (ITLxQ.GE.1) RVxOPTION(6) = RTLx1
                GO TO 20200
      20400     RETURN
      C
      C         Line 20400 is selected when the decides that all necessary
      C           values have been entered and that it is time to start
      C           sending out the rays.
```

```
Line#  Source Line

       C
       20600   IFSxEXI = 1
       C
       C       This sets the EXIT flag to 1, indicating that the user has
       C         decided to exit the program.
       C
               RETURN
       22009   FORMAT(1X,'Enter starting elevation (deg), use neg.')
               END
```

DOXDATA Local Symbols

Name	Class	Type	Size
IFSXEXI	param		
IFSXNEW	param		
IFSXSTA	param		
IVLXTYP	local	INTEGER*4	4
I	local	INTEGER*4	4
J	local	INTEGER*4	4
K	local	INTEGER*4	4
ITLXQ	local	INTEGER*4	4
KVLXFIL	local	CHAR*10	10
KVLXANS	local	CHAR*1	1
IVLXVAR	local	INTEGER*4	4
RVLXLAT	local	REAL*8	8
ITLXTMP	local	INTEGER*4	4
RPXDTOR	local	REAL*8	8
KTLXTMP	local	CHAR*10	10
RVLXVAR	local	REAL*8	8
RVLXLON	local	REAL*8	8
ITLXC1	local	INTEGER*4	4
KVLXPMT	local	CHAR*40	40
ITLXC2	local	INTEGER*4	4
ITLXC3	local	INTEGER*4	4
RTLX1	local	REAL*8	8
RTLX2	local	REAL*8	8
KCLXLBL	local	CHAR*7	42
IVXPARAM	MAINDAT	INTEGER*4	24
IVXLAUN	MAINDAT	INTEGER*4	12
RVXGRID	MAINDAT	REAL*8	48
RVXIONPT	MAINDAT	REAL*8	86400
RVXLAUN	MAINDAT	REAL*8	56
RVXOPTION	MAINDAT	REAL*8	56
RVXION1	MAINDAT	REAL*8	14400
RVXION2	MAINDAT	REAL*8	14400
RVXION3	MAINDAT	REAL*8	14400
RVXION4	MAINDAT	REAL*8	14400
RVXION5	MAINDAT	REAL*8	14400

DOXDATA Local Symbols

Name	Class	Type	Size
RVXION6	MAINDAT	REAL*8	14400

Line# Source Line

```
      SUBROUTINE IOxMENU(N, CHOICE)
C
C-------------------------------------------------------------------
C
C   IOxMENU -- SUBROUTINE TO SET UP A MENU BY READING 'TEMPLATES'
C        FOR THE MENUS FROM FILES, THEN RETURNING THE MENU CHOICE
C
C       CALLED BY: DOxDATA
C
C       CALLS:  IOxPRET
C
C-------------------------------------------------------------------
C
C        AUTHOR:          ERIC L. STROBEL
C
C        DATE:            10/09/86
C
C        VERSION:         1.0
C
C-------------------------------------------------------------------
C
C        REVISED:         10/09/86 -- V1.0.  Initial revision.
C
C-------------------------------------------------------------------
C
C        USES:         N          NUMBER DENOTING WHICH MENU RESOURCE
C                                 TO USE
C
C        TO PRESENT THE USER WITH A MENU AND THEN SEND THE USER'S
C                CHOICE BACK.  (It should be noted that this routine
C                was written before adoption of the variable naming
C                convention.  Due to the short length of the routine
C                it has never seemed worth the change.)
C
C        RETURNS:     CHOICE      THE MENU CHOICE THAT WAS MADE
C
C-------------------------------------------------------------------
C
      INTEGER N, CHOICE, ITEMS, Q
      REAL*8 DUMMY
      CHARACTER*10 FILES(10), FILNAM, KDUMMY
      CHARACTER*40 TITLE, ITEMLIST(10), PROMPT
      CHARACTER*1 CH
C
C ------------------------------------------------------------------
C
C       Read the menu resource from a file set.  The scheme for
C          maintaining menus goes as follows...  The file MASTER.MEN
C          contains a list of the names of the menu files, in the
C          order by which they are numbered in DOxDATA.  The indivi-
```

Line# Source Line

```
C          dual menu files have the following format:
C
C                         # of items, MENU TITLE
C                         Text
C                         of
C                         each
C                         item
C                         (line-by-line, one line per item)
C                         Prompt text
C
C          The existing menu files should be consulted as examples.
C
C ---------------------------------------------------------------------
C
        OPEN(20, FILE='MASTER.MEN', STATUS='OLD')
        READ(20,200) (FILES(I), I=1,10)
200     FORMAT(A10)
        CLOSE(20)
        FILNAM = FILES(N)
        OPEN(21, FILE=FILNAM, STATUS='OLD')
        READ(21,300) ITEMS, TITLE
300     FORMAT(I4,A40)
        READ(21,400) (ITEMLIST(J), J=1, ITEMS)
400     FORMAT(A40)
        READ(21,500) PROMPT
500     FORMAT(A40)
        CLOSE(21)
C
C ------------------------------------------------------------
C
C        Display the menu's list of items.
C
C ------------------------------------------------------------
C
550     WRITE(6,600)
600     FORMAT(/////////////////////////////////////////////////)
        WRITE(6,700) TITLE
700     FORMAT(20X,A40,//)
        WRITE(6,800) (ITEMLIST(J), J=1, ITEMS)
800     FORMAT(5X,A40,/)
C
C ------------------------------------------------------------
C
C        Get the user's response and check it.
C
C ------------------------------------------------------------
C
        CHOICE = 0
900     CALL KEYBRD(PROMPT,1,Q,CHOICE,DUMMY,KDUMMY)
        IF (Q.LT.1) GO TO 900
```

Line#   Source Line

```
          IF (CHOICE.LT.1.OR.CHOICE.GT.ITEMS) THEN
              PRINT *,' INVALID CHOICE.'
              GO TO 550
          ENDIF
          CALL IOxPRET(CH)
          IF (CH.EQ.'A') GO TO 550
          RETURN
          END
```

IOXMENU Local Symbols

Name	Class	Type	Size
CHOICE.	param		
N	param		
TITLE	local	CHAR*40	40
ITEMS	local	INTEGER*4	4
I	local	INTEGER*4	4
J	local	INTEGER*4	4
CH.	local	CHAR*1	1
DUMMY	local	REAL*8	8
Q	local	INTEGER*4	4
KDUMMY.	local	CHAR*10	10
PROMPT.	local	CHAR*40	40
ITEMLIST.	local	CHAR*40	400
FILES	local	CHAR*10	100
FILNAM.	local	CHAR*10	10

Line# Source Line

```
      SUBROUTINE IOxPRET(XXX)
C
C-----------------------------------------------------------------
C
C   IOxPRET -- SUBROUTINE TO PROMPT THE USER TO PRESS RETURN TO
C         CONTINUE, AND THEN CLEAR THE SCREEN
C
C       CALLED BY: MAIN, IOxMENU
C
C       CALLS: KEYBRD
C
C-----------------------------------------------------------------
C
C       AUTHOR:          ERIC L. STROBEL
C
C       DATE:            09/01/87
C
C       VERSION:         2.0
C
C-----------------------------------------------------------------
C
C       REVISED:         10/09/86 -- V1.0.  Initial revision.
C
C                        09/01/87 -- V2.0.  Uses KEYBRD and accepts
C                        a period '.' as a response (for VAX batching
C
C-----------------------------------------------------------------
C
C       This routine displays a 'Press RETURN to continue.' prompt.
C          Upon receiving a RETURN, the screen is cleared.  After
C          two botched attempts, if the user types something out
C          of the ordinary, the routine gently reminds the user to
C          stick to typing RETURN.  IMPORTANT FEATURES:  1) The
C          user may abort by typing 'A' (for abort) in response to
C          the 'Press ...' prompt.  This provides a mechanism to
C          allow the user to have a chance to avoid doing something
C          irrevocable if the choice was made by mistake.  2) In
C          an effort to allow the program to be run under VMS in
C          batch mode, a response of '.' & RETURN has the same effect
C          as just typing RETURN.  For whatever reason, a VMS batch
C          file cannot contain a blank line.
C
C       Please note that this routine was largely written before the
C          adoption of the variable naming convention used throughout
C          most of the rest of the program.  This routine is brief
C          enough that it has proven so far unnecessary to change it.
C
C ----------------------------------------------------------------
C
      CHARACTER*40 PRMT
```

```
Line#  Source Line

              CHARACTER*10 CHTMP
              CHARACTER*1 XXX
              INTEGER YYY, ZZZ, ITMP, ITYPE
              REAL*8 DTMP
              XXX = ' '
        50    PRMT = ' Press RETURN to continue. '
              CALL KEYBRD(PRMT,3,YYY,ITMP,DTMP,XXX)
              IF (YYY.LT.1.OR.XXX.EQ.'A'.OR.XXX.EQ.'.') THEN
                  WRITE(6,300)
       300        FORMAT(//////////////////////////////////////////////////////
                  RETURN
              ELSE
                  ZZZ = ZZZ + 1
                  IF (ZZZ.GT.2) PRINT *,' HIT RETURN ONLY TO CONTINUE.'
                  GO TO 50
              ENDIF
              RETURN
              END
```

IOXPRET Local Symbols

Name	Class	Type	Size
XXX	param		
PRMT.	local	CHAR*40	40
YYY	local	INTEGER*4	4
ZZZ	local	INTEGER*4	4
DTMP.	local	REAL*8	8
ITMP.	local	INTEGER*4	4

Line# Source Line

```
      SUBROUTINE KEYBRD(PSTR,ITYP,LN,IVAR,FVAR,CVAR)
C
C-------------------------------------------------------------
C
C   KEYBRD -- SUBROUTINE TO TAKE KEYBRD INPUT INTELLIGENTLY AND
C         PARSE IT.
C
C         CALLED BY: MAIN, DOxDATA, IOxPRET
C
C-------------------------------------------------------------
C
C         AUTHOR:          ERIC L. STROBEL
C
C         DATE:            09/01/87
C
C         VERSION:         1.0
C
C-------------------------------------------------------------
C
C         REVISED:         09/01/87 -- V1.0.  Initial revision.
C
C-------------------------------------------------------------
C
C         USES:        PSTR         The prompt string.
C
C                      ITYP         The type of answer expected to be
C                                  returned.  1 = INT, 2 = FLOAT.
C                                  3 = CHAR.
C
C
C         To present the user with a prompt and to discern what
C              sort of response is expected.  This routine attempts
C              to be somewhat intelligent about parsing the typed
C              response.  Real values entered without a decimal
C              point are actually read correctly, and strings are
C              converted to all CAPS so that the response typed in
C              appears to be case insensitive.  This is a modification
C              of a routine authored by Frank Rhoads.  Please note the
C              comment block below refering to where changes need to
C              be made in converting from system to system.
C
C
C         RETURNS:     LN          The number of characters typed in
C                                  the keyboard.
C
C                      IVAR         The integer value to be passed.
C
C                      FVAR         The floating value to be passed.
C
C                      CVAR         The character string to be passed.
```

```
Line#  Source Line

C
C-----------------------------------------------------------------
C
           CHARACTER*(*)CVAR, PSTR
           CHARACTER*20 CVR
           CHARACTER*8 IFMT
           CHARACTER*1 BYTS(20)
           EQUIVALENCE(BYTS,CVR)
           INTEGER*4 IVAR,LN
           REAL*8 FVAR
           CVR='                    '
    9      LN=0
C
C -----------------------------------------------------------------
C     Display the appropriate prompt, including a default value.
C
C     This is also where the changes must be made in translating
C         from system to system.  In the three WRITEs below, the
C         formatting ends with ... " ,2H):" and then whatever is
C         the appropriate thing to suppress the return to the
C         beginning of the next line.  On the VAX (VMS) this is
C         ",S".  With Microsoft Fortran (DOS) it is ",\", and with
C         RyanMcFarland Fortran (DOS) it is nothing at all.
C -----------------------------------------------------------------
C
           IF (ITYP.EQ.1) THEN
                WRITE(6,'(1X,A40,1H(,I4,2H):,\)') PSTR, IVAR
           ELSE IF (ITYP.EQ.2) THEN
                WRITE(6,'(1X,A40,1H(,G20.9,2H):,\)') PSTR, FVAR
           ELSE
                WRITE(6,'(1X,A40,1H(,A10,2H):,\)') PSTR, CVAR
           ENDIF
C
C     Get the user's response as a character string.
C
           READ(5,'(A20)')CVR
C
C     Find the length of the response.
C
           DO 10 I=1,20
             L=21-I
             IF(BYTS(L).NE.' ')GO TO 11
   10      CONTINUE
           L=0
   11      LN=L
C
C     No response, so Return key must have been typed.
C
           IF(LN.EQ.0)RETURN
           CVAR=CVR
```

Line# Source Line

```
              IF(ITYP.LT.3)THEN
C
C       If the expected answer is a numeric type, then read it
C          from the internal string.
C
                IF(ITYP.EQ.1)THEN
                  WRITE(IFMT,'(2H(I,I2,1H))')LN
                  READ(CVR,IFMT,ERR=12)IVAR
                ELSE
                  WRITE(IFMT,'(2H(F,I2,3H.0))')LN
                  READ(CVR,IFMT,ERR=12)FVAR
                ENDIF
              ELSE
C
C       For a character string answer, convert to all caps.
C
   20           DO 21 ICI=1,LN
                  K=Ichar(BYTS(ICI))
                  If(k.GE.96)THEN
                    K=K-32
                    CVAR(ICI:ICI)=Char(K)
                  ELSE
                    CVAR(ICI:ICI)=BYTS(ICI)
                  ENDIF
   21           CONTINUE
              ENDIF
              RETURN
   12         WRITE(6,'(37H Conversion ERROR, Please RETRY Input)')
              GO TO 9
  100         RETURN
              END
```

KEYBRD Local Symbols

Name	Class	Type	Size
CVAR.	param		
FVAR.	param		
IVAR.	param		
LN.	param		
ITYP.	param		
PSTR.	param		
I	local	INTEGER*4	4
K	local	INTEGER*4	4
L	local	INTEGER*4	4
ICI	local	INTEGER*4	4
CVR	local	CHAR*20	20
IFMT.	local	CHAR*8	8
BYTS.	local	CHAR*1	20

```
Global Symbols

Name                          Class     Type             Size

DOXDATA . . . . . . . . . . . FSUBRT    ***              ***
IOXMENU . . . . . . . . . . . FSUBRT    ***              ***
IOXPRET . . . . . . . . . . . FSUBRT    ***              ***
KEYBRD. . . . . . . . . . . . FSUBRT    ***              ***
MAINDAT . . . . . . . . . . . common    ***              86596
RAYSUB. . . . . . . . . . . . extern    ***              ***
RESULTS . . . . . . . . . . . common    ***              1320
main. . . . . . . . . . . . . FSUBRT    ***              ***

Code size = 56db (22235)
Data size = 0a16 (2582)
```

Line# Source Line

```
      SUBROUTINE RAYSUB(RVxAZI, RVxELEV)
C
C ---------------------------------------------------------------
C
C   RAYSUB  --   3D RAYTRACING WITH ACCELERATED FREE SPACE
C                 PROPAGATION CALCULATION
C
C   CALLED BY:     MAIN
C
C   CALLS:         Almost everything else, as this is really
C                  the core of the program.
C
C ---------------------------------------------------------------
C
C   AUTHORS:      MICHAEL H. REILLY & ERIC L. STROBEL
C
C   DATE:         03/18/88
C
C   VERSION:      4.3
C
C ---------------------------------------------------------------
C
C   REVISED:      07/25/86 -- Initial revision.  Translated
C                 from Tektronix Basic to VAX Fortran by
C                 Eric L. Strobel.
C
C                 07/30/86 -- V1.1.  Change over to use of
C                 REAL*8 precision in the calculations.
C
C                 08/12/86 -- V2.0.  Add the capability to
C                 perform propagation loss calculations.
C                 Also greatly massaged over the output.
C                 A number of minor and MAJOR bugs corrected.
C
C                 09/05/86 -- V2.3.  Add horizon focusing and
C                 handle low angle rays.  Record data for
C                 proper handling of skip focusing.  Record
C                 maximum height.  Calculate deviation from
C                 great circle path.  Minor bug fixes.
C
C                 09/17/86 -- V2.4.  Fix calculation of great
C                 circle path deviation.  Introduce cutoff in
C                 number of bounces.  Automagically obtain
C                 approximate bottom of the ionosphere, elim-
C                 inating the need for the prompt.  Also fixed
C                 rotation matrix used in calculating the new
C                 c-values.
C
C                 10/10/86 -- V3.0.  Former program, now demoted
C                 to a subroutine.  Appropriate changes made.
```

Line# Source Line

```
C                 Duties performed by subroutine GEC now done
C                 elsewhere.  Portion of IONOPAR that searches
C                 for triangles of points now streamlined so
C                 that the search won't outlast the Earth!
C
C                 09/01/87 -- V4.0.  The essentials have been
C                 listed in the comment block at the beginning
C                 of MAIN (RAYTRA4.FOR), so you can read about
C                 the changes there.
C
C                 09/30/87 -- V4.1.  Phase path calcs. added.
C
C                 01/15/88 -- V4.2.  The (incorrect) derivative
C                 values are no longer reported.  One of the rays
C                 in the bundle has been deleted, for speed's sake.
C                 The range cutoff has been tweaked so that it will
C                 occur nearly exactly, not just at the end of the
C                 next increment.  Some of the other changes have
C                 been discussed in the comment block at the
C                 beginning of MAIN (RAYTRA4.FOR).
C
C                 03/18/88 -- V4.3.  Corrections made for angular range.
C                 (see comment block at beginning of MAIN)  In the main
C                 body of this routine, the azimuth is the running
C                 value from the launch point to the previous raypoint.
C                 The C's are now normalized at every step, which should
C                 enhance the accuracy.  Some other minor tweaking.
C
C ---------------------------------------------------------
C
C       USES:     RVxAZI          Ray launch azimuth
C
C                 RVxELEV         Ray launch elevation
C
C                 /MAINDAT/       A common block the ionospheric
C                                  and user data that was read in
C                                  the previous routines.
C
C              To perform 3-D raytracing of radio propaga-
C         tion through the ionosphere.  Incorporates a spec-
C         ific model for true height profiles of electron
C         density.  Does propagation loss due to focusing and
C         to reflection from the ground.  Doesn't (yet) incorporate
C         magnetic field effects, or deviative absorption effects.
C
C       RETURNS:  RVxBOUN        The array of results at the
C                                  earth impact and ray end points.
C
C ---------------------------------------------------------
C
```

Line# Source Line

```
            INTEGER IVxPARAM(6), IVxLAUN(3)
            INTEGER IFxBUN, IVxSSNUM, IVxTIME(5), IVxNBOU, IFxGRND
            INTEGER IVxSCSING, IVxSCTRI1, IVxSCTRI2, IVxSCTRI3
            INTEGER ICxNSCP, IVxJ, IVxSX, IVxSY, IVxSZ, IFxN4, IFxGEN
            INTEGER IVxNBL, IFxSFL, ITxK, IFxEND
            INTEGER IFxPPF, IFxCASE, IFxCCAL
C
            LOGICAL LVxRCUT, LVxHCUT, LVxEND
C
            REAL*8 RVxGRID(6), RVxIONPT(30,60,6), RVxLAUN(7)
            REAL*8 RVxOPTION(7), RVxLA, RVxLO, RVxDD6
            REAL*8 RVxBNDL(4,3,11), RVxBOUN(11,15), RVxELEVO
            REAL*8 RVxAZIO, RVxYEAR, RVxA100, RVxTIM
            REAL*8 RPxPI,RPxDTOR,RPxREARTH,RVxLATSC(1800),RVxLONSC(1800)
            REAL*8 RVxFNSQ(1800,3), RVxH(1800,3), RVxSEZGEC(3,3)
            REAL*8 RVxFNSB(3), RVxHB(3), RVxCOFMAT(2,2), RVxLATO
            REAL*8 RVxLONO, RVxHBOT, RVxRALIM, RVxANGLIM, RVxFREQ
            REAL*8 RVxFSQU, RVxELEV, RVxAZI, RVxRPI, RVxCX, RVxCY
            REAL*8 RVxCZ, RVxSINEL, RVxXI, RVxYI, RVxZI, RVxH0, RVxH5
            REAL*8 RVxGPLTOT, RVxDDSZ1, RTxDZCALC, RVxDELZ, RVxHZ
            REAL*8 RVxXF, RVxYF, RVxZF, RTxS4, RVxGPLINC, RVxDDXHB
            REAL*8 RVxDDYHB, RVxUDXDS, RVxUDYDS
            REAL*8 RVxUDZDS, RVxDXDZ, RVxDYDZ, RVxXGRAD, RVxYGRAD
            REAL*8 RVxDET, RTxP1, RTxP2, RTxP3, RVxALPHA
            REAL*8 RVxBETA, RVxETA1, RVxETA2, RVxLATI, RVxLONI
            REAL*8 RVxLAT1, RVxLON1, RVxX7, RVxY7, RVxZ7
            REAL*8 RVxR1, RVxR2, RVxB5(3), RVxB6(3), RVxA5(3), RVxA6(3)
            REAL*8 RVxE5(3), RVxE6(3), RVxV1, RVxV2
            REAL*8 RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG
            REAL*8 RVxDR, RVxD2R, RVxDEV, RVxCSUM
            REAL*8 RVxHCUT, RPxHTOP, RVxHMIN, RTxTEMP
            REAL*8 RVxXX, RVxHBND, RVxPPL, RVxPPI, RVxMU, RVxMU2
            REAL*8 RVxF40, RVxF65, RVxK1, RVxHL, RVxCOSEL
            REAL*8 RVxRTMP, RVxRMAX, RVxDELR, RVxDELR2
            REAL*8 RTxAINC, RTxRT1
C
C------------------------------------------------------------------
C
C     NOTE:  IVxTIME(1) = YEAR; IVxTIME(2) = MONTH
C            IVxTIME(3) = DAY;  IVxTIME(4) = HOUR
C            IVxTIME(5) = MINUTE
C
C            RVxBNDL = RVxBNDL( RAY 1,2,3,OR 4; 1-LAT,2-LON,3-ALT;
C                               # OF BOUNCES)
C
C            RVxBOUN = RVxBOUN(N-TH BOUNCE; PARAMS),
C         PARAMS--           1 -> LAT
C                            2 -> LON
C                            3 -> GROUP PATH LENGTH
C                            4 -> ANGULAR RANGE
```

Line# Source Line

```
C                              5 -> DB DOWN
C                              6 -> DEVIATION FROM GR. CIRCLE PATH
C                              7 -> UNUSED
C                              8 -> UNUSED
C                              9 -> HEIGHT OF THE DATA POINT
C                             10 -> PHASE PATH LENGTH
C                             11 -> X DIRECTION COSINE OF RAY
C                             12 -> Y DIRECTION COSINE OF RAY
C                             13 -> Z DIRECTION COSINE OF RAY
C                          14,15 -> RESERVED FOR FUTURE USE
C
C-----------------------------------------------------------------
C
      COMMON /MAINDAT/ RVxGRID, RVxIONPT, IVxPARAM, IVxLAUN
     #,RVxLAUN ,RVxOPTION
      COMMON /RESULTS/ RVxBOUN
      COMMON /LPARM/ IVxSSNUM,IVxTIME,RVxYEAR,RVxA100,IFxGRND
     *,RVxTIM
      COMMON /LOSSES/ RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG
      COMMON /PRAM/ RPxPI, RPxDTOR, RPxREARTH, RPxHTOP
      COMMON /SCPS1/ ICxNSCP, RVxLATSC, RVxLONSC
      COMMON /SCPS1A/ RVxFNSQ,RVxH
      COMMON /OTHER/ RVxLAT1,RVxLON1,RVxHBOT,RVxFSQU,RVxRPI,RVxHCUT
      COMMON /MISC/ RVxSEZGEC,RVxX7,RVxY7,RVxZ7,RVxR1,RVxR2,RVxHMIN
      COMMON /START/ RVxXI, RVxYI, RVxZI, RVxLATI, RVxLONI
      COMMON /END/ RVxXF, RVxYF, RVxZF, RVxH5
      COMMON /IONO1/ RVxALPHA, RVxBETA,RVxETA1,RVxETA2,RVxFNSB,RVxHB
      COMMON /IONO2/ RVxA5, RVxA6, RVxB5, RVxB6, RVxE5, RVxE6
      COMMON /IONO3/ RVxV1,RVxV2,RVxXX,IFxCASE, RVxHBND
      COMMON /MORE/ IVxSX, IVxSY, IVxSZ, RVxCX, RVxCY, RVxCZ
      COMMON /TEMP1/ RVxF40, RVxF65, RVxK1, RVxHL
      COMMON /GORP/ RVxDD6, RVxANGLIM
C
C       Initialize things
C
      DO 9000 I=1,5
          IVxTIME(I) = IVxPARAM(I+1)
 9000 CONTINUE
      RPxPI = 3.141592654D00
      RPxDTOR = 0.0174532925D00
      RPxREARTH = 6371.2D00
      RPxHTOP = 2000.0D00
      RVxHMIN = RVxLAUN(7)
      ITxK = 0
C
C     Initialize some things that are specific to the RADAR-C
C       model.
C
      RVxHBOT = 40.0D00
      RVxF40 = 4.03D-04
```

Line# Source Line

```
          RVxK1 = 0.12D00
          RVxF65 = RVxF40*DEXP(25.0D00 * RVxK1)
          RVxHL = 120.48384357
C
C   Begin input section, i.e. break out the arrays that will
C    be needed from the large array.
C
10000     ICxNSCP = IDNINT(RVxGRID(5)*RVxGRID(6))
          DO 10200 I=1, RVxGRID(5)
              RVxLA = RVxGRID(3) + (I-1)*RVxGRID(1)
              DO 10100 J = 1, RVxGRID(6)
                  RVxLO = RVxGRID(4) + (J-1)*RVxGRID(2)
                  ITxK = ITxK + 1
                  RVxLATSC(ITxK) = RVxLA
                  RVxLONSC(ITxK) = RVxLO
                  RVxFNSQ(ITxK,1) = RVxIONPT(I,J,1)
                  RVxFNSQ(ITxK,2) = RVxIONPT(I,J,3)
                  RVxFNSQ(ITxK,3) = RVxIONPT(I,J,6)
                  RVxH(ITxK,1) = RVxIONPT(I,J,2)
                  RVxH(ITxK,2) = RVxIONPT(I,J,5)
                  RVxH(ITxK,3) = RVxIONPT(I,J,4)
                  RVxLATSC(ITxK) = RVxLATSC(ITxK) * RPxDTOR
                  RVxLONSC(ITxK) = RVxLONSC(ITxK) * RPxDTOR
10100         CONTINUE
10200     CONTINUE
C
C          More initializing.
C
          RVxLAT0 = RVxLAUN(1)
          RVxLON0 = RVxLAUN(2)
          RVxLAT0 = RVxLAT0 * RPxDTOR
          RVxLON0 = RVxLON0 * RPxDTOR
          RVxRALIM = RVxLAUN(3)
          RVxANGLIM = RVxRALIM/RPxREARTH
          RTxS4 = 0.0D00
          IVxSSNUM = IVxPARAM(1)
          IFxGRND = IVxLAUN(3)
          IFxPPF = 1
          IVxNBL = IVxLAUN(1)
          IF (IVxNBL.GT.10) IVxNBL = 10
          IF (IVxNBL.LT.0) IVxNBL = 0
          IFxSFL = IVxLAUN(2)
          RVxFREQ = RVxLAUN(5)
          RVxFSQU = RVxFREQ * RVxFREQ
          RVxRPI = RVxLAUN(4)
          RVxHCUT = RVxLAUN(7)
          IF (RVxELEV.EQ.0.0D00) RVxELEV = 0.01D00
C
C   BEGIN CALCULATIONAL SECTION
C
```

Line# Source Line

```
20000   RVxELEV = RVxELEV * RPxDTOR
        RVxAZI = RVxAZI * RPxDTOR
        RVxELEV0 = RVxELEV
        RVxAZI0 = RVxAZI
        IVxJ = 0
        IFxCCAL = 0
        RVxDD6 = RVxRPI
        RVxLOSR = 0.0D00
        RVxLOSG = 0.0D00
        RVxLOSA = 0.0D00
20020   IFxBUN = 3
        IF (IFxSFL.EQ.1) IFxBUN = 0
C
C        Determine the elevation & azimuth for each ray in the bundle.
C
        RTxAINC = 0.00125D00 * RPxDTOR
20050   IF (IFxBUN.EQ.3) THEN
            RVxAZI = RVxAZI0 + RTxAINC
            RVxELEV = RVxELEV0 + RTxAINC
        ELSE IF (IFxBUN.EQ.2) THEN
            RVxAZI = RVxAZI0 + RTxAINC
            RVxELEV = RVxELEV0 - RTxAINC
        ELSE IF (IFxBUN.EQ.1) THEN
            RVxAZI = RVxAZI0 - RTxAINC
            RVxELEV = RVxELEV0 - RTxAINC
        ELSE
            RVxELEV = RVxELEV0
            RVxAZI = RVxAZI0
        ENDIF
C
C        Still more initializing.
C
        IVxNBOU = 0
        IVxJ = 0
        RVxLAT1 = RVxLAT0
        RVxLON1 = RVxLON0
        RVxH0 = RVxLAUN(6)
        RVxXI = 0.0D00
        RVxYI = 0.0D00
        RVxZI = RVxH0
        IVxSZ = INTSIGN(DSIN(RVxELEV))
        RVxH5 = RVxH0
        RTxTEMP = 0.0D00
C
C        Do the initial coordinate setup and transformations.
C
        CALL ROTSEZ
        CALL LATLON3
        IFxN4 = 1
        IFxGEN = 1
```

```
Line#  Source Line

C
C        Get the initial values for the ionospheric parameters
C          and (if any) gradients.
C
        CALL IONOPAR(IVxJ, IFxN4, IFxGEN, IVxSCSING, IVxSCTRI1
     *, IVxSCTRI2, IVxSCTRI3)
        RVxMU2 = 1.0D00 - RVxXX/RVxFSQU
        IF (RVxMU2.LT.0.0D00) THEN
            PRINT *, ' RAY DOES NOT PROPAGATE!!!!'
            STOP
        ENDIF
        RVxCOSEL = DCOS(RVxELEV)
        RVxSINEL = DSIN(RVxELEV)
        RVxMU = DSQRT(RVxMU2)
        RVxCX = -RVxMU * RVxCOSEL * DCOS(RVxAZI)
        IVxSX = INTSIGN(RVxCX)
        RVxCY = RVxMU * RVxCOSEL * DSIN(RVxAZI)
        IVxSY = INTSIGN(RVxCY)
        RVxCZ = RVxMU * RVxSINEL
        IVxSZ = INTSIGN(RVxCZ)
        RVxCX = RVxCX * RVxCX
        RVxCY = RVxCY * RVxCY
        RVxCZ = RVxCZ * RVxCZ
        RVxPPL = 0.0D00
        RVxGPLTOT = 0.0D00
        IVxIII = 0
C
        LVxEND = .FALSE.
        LVxHCUT = .FALSE.
        LVxRCUT = .FALSE.
C
C        Begin the ray increment by estimating the z increment
C          for the upcoming raypath increment.
C
20106   RVxDDSZ1 = IVxSZ * DSQRT( RVxCZ / (RVxCX + RVxCY + RVxCZ) )
        IF (RVxALPHA.NE.0.0D00.AND.RVxBETA.NE.0.0D00) THEN
C
C       In this case, the index of refraction is quadratic in z, so
C         an additional correction is appropriate.
C
            RTxDZCALC = (RVxDDSZ1 - RVxSINEL) / RVxDD6
        ELSE
            RTxDZCALC = 0.0D00
        ENDIF
        RVxDELZ = IVxSZ * RVxRPI * (RVxDDSZ1 + RTxDZCALC*RVxRPI/2.0D00)
        RVxDELZ = DABS(RVxDELZ)
C
C       This section is here to ensure that the calculation cuts off
C         in range almost exactly when it is supposed to.
C
```

Line# Source Line

```
            RVxDELR = DSQRT(RVxRPI*RVxRPI - RVxDELZ*RVxDELZ)
            RVxRTMP = (RTxS4 + RVxDELR/(RVxH0+RPxREARTH)) * RPxREARTH
            RVxRMAX = RVxANGLIM * RPxREARTH
            IF (RVxRTMP.GE.RVxRMAX) THEN
                RVxDELR2 = RVxDELR - ((RVxRTMP-RVxRMAX)*(RVxH0+RPxREARTH)
     #/RPxREARTH)
                RVxDELZ = RVxDELZ * RVxDELR2 / RVxDELR
            ENDIF
C
C           The angle of ray incidence at H = 100 km.  This is used in the
C             (currently commented out) absorption loss calculation.
C
C           IF (DABS(RVxHZ - 100.0D00).LT.(RVxRPI/2.0D00)) RVxA100 =
C    #DCOS(RVxDELZ/RVxRPI)
C
            RVxDELZ = DMAX1( RVxDELZ, 1.0D-04)
            IFxGEN = 1
            RVxHZ = RVxH0 + IVxSZ*RVxDELZ
C
C           Check to see if a boundary has been crossed.
C
            IF (IVxSZ.LT.0) THEN
                IF (RVxHZ.LT.RVxHBND) THEN
                    RVxDELZ = RVxH0 - RVxHBND
                    IFxGEN = 2
                ENDIF
                GO TO 20133
            ENDIF
            IF (RVxHZ.GT.RVxHBND) THEN
                RVxDELZ = RVxHBND - RVxH0
                IFxGEN = 2
            ENDIF
C
20133       RVxHZ = RVxH0 + IVxSZ*RVxDELZ
C
C            Use the z increment and height to calculate where the
C             ray ends up at.
C
            CALL ENDPT(IFxGEN, RVxH0, RVxHZ, RTxS4, RVxGPLINC)
            RVxHZ = RVxH0 + RVxZF
            RTxRT1 = RPxREARTH + RVxHZ
            RVxH5 = DSQRT(RVxXF*RVxXF + RVxYF*RVxYF + RTxRT1*RTxRT1)
     #- RPxREARTH
            IF (RVxH5.LT.0.0D00) RVxH5 = 0.0D00
            IF (RVxH5.LT.RVxHMIN) RVxHMIN = RVxH5
C
C            If desired, calculate the phase path increment.  The
C             value of IFxPPF is currently 'hard-wired' to be 1.
C
            IF (IFxPPF.EQ.1) CALL PHSPL(IFxGEN, RVxPPI)
```

Line#   Source Line

```
          IVxIII = IVxIII + 1
          IFxCCAL = 0
          IF (IFxGEN.NE.6.OR.RVxH5.GT.RVxHBOT) GO TO 20150
C
C          Will be FALSE only if the ray is below the bottom of
C           the ionosphere & headed downward.
C
          RVxZI = RVxHZ
          RVxXI = RVxXF
          RVxYI = RVxYF
          CALL LATLON3
          CALL NEWCS(IFxGEN)
          IFxCCAL = 1
C
C     SMALL SECTION TO HANDLE EARTH BOUNCE
C
          CALL ANRANG(RVxAZI,RVxLATI,RVxLONI,RVxLATO,RVxLONO,RTxS4)
          IVxNBOU = IVxNBOU + 1
C
C          Check to see if the cutoff in the number of bounces is met.
C
          IF (IVxNBOU.GT.IVxNBL) THEN
              IVxNBOU = IVxNBOU - 1
              LVxEND = .TRUE.
              GO TO 20150
          ENDIF
C
C ----------------------------------------------------------------
C          If this is the primary ray in the bundle, then all of the
C           information about the bounce point needs to be recorded,
C           otherwise only the parts that will be needed later are
C           stored.
C ----------------------------------------------------------------
C
          IF (IFxBUN.EQ.0) THEN
              RVxBOUN(IVxNBOU,1) = RVxLATI
              RVxBOUN(IVxNBOU,2) = RVxLONI
              RVxBOUN(IVxNBOU,3) = RVxGPLTOT + RVxGPLINC
              RVxBOUN(IVxNBOU,4) = RTxS4
              IF (IFxSFL.NE.1) THEN
C
C         TIMES is a routine which calculates some quantities for the
C           (currently commented out) absorption loss calculation.
C
C                 CALL TIMES(RVxBOUN, IVxNBOU, RVxLONO)
                  CALL LOSS(RVxBOUN,RVxBNDL,IVxNBOU,RVxLATO,RVxLONO,
     #RVxELEVO,IFxEND)
              ENDIF
              RVxBOUN(IVxNBOU,5) = RVxLOSR + RVxLOSG
              CALL GCDEV(IVxNBOU,RVxAZIO,RVxLATO,RVxLONO,RVxBOUN,RVxDEV)
```

Line# Source Line

```
                RVxBOUN(IVxNBOU,6) = RVxDEV
                RVxBOUN(IVxNBOU,9) = RVxH5
                IF (IFxPPF.EQ.1) RVxBOUN(IVxNBOU,10) = RVxPPL + RVxPPI
                RVxCSUM = RVxCX + RVxCY + RVxCZ
                RVxBOUN(IVxNBOU,11) = IVxSX*DSQRT(RVxCX)/DSQRT(RVxCSUM)
                RVxBOUN(IVxNBOU,12) = IVxSY*DSQRT(RVxCY)/DSQRT(RVxCSUM)
                RVxBOUN(IVxNBOU,13) = IVxSZ*DSQRT(RVxCZ)/DSQRT(RVxCSUM)
            ELSE
                RVxBNDL(IFxBUN,1,IVxNBOU) = RVxLATI
                RVxBNDL(IFxBUN,2,IVxNBOU) = RVxLONI
                RVxBNDL(IFxBUN,3,IVxNBOU) = RVxH5
            ENDIF
C
C   END EARTH BOUNCE HANDLING
C
C ---------------------------------------
C
C   Accumulate the group and phase path lengths
C
20150     RVxGPLTOT = RVxGPLTOT + RVxGPLINC
          IF (IFxPPF.EQ.1) RVxPPL = RVxPPL + RVxPPI
C
C   TAKE CARE OF BOUNDARY CROSSING
C
C --------------------------------
C
C        First check for the existence of a boundary crossing.
C
          IF (IFxGEN.NE.2) GO TO 30000
C
C        If this is a spherically symmetric case, there are no
C         tilts, so go on.
C
          IF (IFxN4.EQ.1) GO TO 30000
C
C        Call the routine that will actually determine the spatial
C         tilt of the boundary.
C
          CALL TILTS(RVxDDXHB, RVxDDYHB, IVxJ)
C
C ----------------------------------------------------------
C        Now, we attempt to correct this particular raypath increment
C         to insure that it ends exactly ON the boundary, and doesn't
C         overshoot it.
C ----------------------------------------------------------
C
          RVxUDXDS = IVxSX * DSQRT(RVxCX - RVxETA1*RVxXF)
          RVxUDYDS = IVxSY * DSQRT(RVxCY - RVxETA2*RVxYF)
          RVxUDZDS = IVxSZ * DSQRT(RVxCZ - RVxALPHA*RVxZF + RVxBETA*RVxZF*
     *RVxZF)
```

Line# Source Line

```
          RVxDXDZ = RVxUDXDS / RVxUDZDS
          RVxDYDZ = RVxUDYDS / RVxUDZDS
          RVxXGRAD = RVxDDXHB
          RVxYGRAD = RVxDDYHB
          RVxCOFMAT(1,1) = 1.0D00 - RVxDXDZ * RVxXGRAD
          RVxCOFMAT(1,2) = -RVxDXDZ * RVxYGRAD
          RVxCOFMAT(2,1) = -RVxDYDZ * RVxXGRAD
          RVxCOFMAT(2,2) = 1.0D00 - RVxDYDZ * RVxYGRAD
          RVxDET = RVxCOFMAT(1,1)*RVxCOFMAT(2,2) - RVxCOFMAT(1,2)*
     *RVxCOFMAT(2,1)
          RTxP1 = (RVxXF*RVxCOFMAT(2,2) - RVxYF*RVxCOFMAT(1,2))/RVxDET
          RTxP2 = (RVxCOFMAT(1,1)*RVxYF - RVxCOFMAT(2,1)*RVxXF)/RVxDET
          RTxP3 = RVxZF + RVxXGRAD*RTxP1 + RVxYGRAD*RTxP2
          RVxXF = RTxP1
          RVxYF = RTxP2
C
C         Now, correct the group and phase path length totals.
C
          RVxGPLTOT = RVxGPLTOT + (RTxP3 - RVxZF)/RVxUDZDS
          IF (IFxPPF.EQ.1) THEN
              RVxPPL = RVxPPL + (RVxUDXDS*RVxUDXDS + RVxUDYDS*RVxUDYDS
     #+ RVxUDZDS*RVxUDZDS) * (RTxP3 - RVxZF)/RVxUDZDS
          ENDIF
C
          RVxZF = RTxP3
          RVxHZ = RVxHO + RVxZF
          RTxRT1 = RPxREARTH + RVxHZ
          RVxH5 = DSQRT(RVxXF*RVxXF + RVxYF*RVxYF + RTxRT1*RTxRT1)
     #- RPxREARTH
          IF (RVxH5.LT.0.0D00) RVxH5 = 0.0D00
C
C    BEGIN UPDATE FOR NEXT ITERATION.
C
30000     RVxXI = RVxXF
          RVxYI = RVxYF
          RVxZI = RVxHZ
30003     CALL LATLON3
          LVxHCUT = (RVxH5.GE.RVxHCUT.AND.IVxSZ.GT.0)
          IF (IFxCCAL.EQ.0.OR.LVxHCUT) CALL NEWCS(IFxGEN)
C
C         If it is the end of the problem, there's a bunch of stuff
C           that doesn't need to be done, so skip it.
C
          IF (LVxHCUT.OR.LVxEND) GO TO 30300
30203     CALL IONOPAR(IVxJ, IFxN4, IFxGEN, IVxSCSING, IVxSCTRI1
     *, IVxSCTRI2, IVxSCTRI3)
C
C          Renormalize the c-values in order to reduce the errors that
C            might otherwise propagate through the program.
C
```

```
Line#  Source Line

                RVxCSUM = RVxCX + RVxCY + RVxCZ
                RVxMU2 = 1.0D00 - RVxXX/RVxFSQU
                RVxCX = RVxCX * RVxMU2 / RVxCSUM
                RVxCY = RVxCY * RVxMU2 / RVxCSUM
                RVxCZ = RVxCZ * RVxMU2 / RVxCSUM
       C
                RVxH0 = RVxH5
                RVxLAT1 = RVxLATI
                RVxLON1 = RVxLONI
                RVxSINEL = RVxDDSZ1
                CALL ROTSEZ
       30300    CALL ANRANG(RVxAZI,RVxLATI,RVxLONI,RVxLAT0,RVxLON0,RTxS4)
       C
       C        Evaluate the problem cutoff criteria.
       C
                LVxRCUT = (RTxS4.GE.RVxANGLIM)
                LVxHCUT = (RVxH5.GE.RVxHCUT.AND.IVxSZ.GT.0)
       C         WRITE(70,30301) RTxS4*RPxREARTH, RVxH5
       C30301    FORMAT(1X,G18.8,G18.8)
       C
       C         Check for whether the ray has reached a cutoff condition.
       C
                IF (LVxRCUT.OR.LVxHCUT.OR.LVxEND) THEN
       C
       C -----------------------------------------------------------
       C         A cutoff criterion for this problem has been met.  If it
       C          is not the primary ray, record the necessary numbers and
       C          re-cycle for the next ray in the bundle, otherwise,
       C          record all the information on the endpoint and return to
       C          MAIN.
       C -----------------------------------------------------------
       C
                    IF (IFxBUN.NE.0) THEN
                        RVxBNDL(IFxBUN,1,IVxNBOU+1) = RVxLATI
                        RVxBNDL(IFxBUN,2,IVxNBOU+1) = RVxLONI
                        RVxBNDL(IFxBUN,3,IVxNBOU+1) = RVxH5
                        IFxBUN = IFxBUN - 1
                        RVxHMIN = RVxLAUN(7)
                        GO TO 20050
                    ENDIF
                    IFxEND = 1
                    RVxLAT1 = RVxLATI
                    RVxLON1 = RVxLONI
                    CALL ROTSEZ
                    IVxNBOU = IVxNBOU + 1
                    RVxBOUN(IVxNBOU,1) = RVxLATI
                    RVxBOUN(IVxNBOU,2) = RVxLONI
                    RVxBOUN(IVxNBOU,3) = RVxGPLTOT
                    RVxBOUN(IVxNBOU,4) = RTxS4
                    IF (IFxSFL.NE.1) THEN
```

```
Line#  Source Line

       C
       C        TIMES is called to calculate some values needed by the
       C          (currently commented out) absorption loss calculation.
       C
       C                CALL TIMES(RVxBOUN, IVxNBOU, RVxLONO)
                        CALL LOSS(RVxBOUN,RVxBNDL,IVxNBOU,RVxLATO,RVxLONO,
           #RVxELEVO,IFxEND)
                    ENDIF
                    RVxBOUN(IVxNBOU,5) = RVxLOSG + RVxLOSR
                    CALL GCDEV(IVxNBOU,RVxAZIO,RVxLATO,RVxLONO,RVxBOUN,RVxDEV)
                    RVxBOUN(IVxNBOU,6) = RVxDEV
                    RVxBOUN(IVxNBOU,9) = RVxH5
                    RVxBOUN(IVxNBOU,10) = RVxPPL
                    RVxCSUM = RVxCX + RVxCY + RVxCZ
                    RVxBOUN(IVxNBOU,11) = IVxSX*DSQRT(RVxCX)/DSQRT(RVxCSUM)
                    RVxBOUN(IVxNBOU,12) = IVxSY*DSQRT(RVxCY)/DSQRT(RVxCSUM)
                    RVxBOUN(IVxNBOU,13) = IVxSZ*DSQRT(RVxCZ)/DSQRT(RVxCSUM)
                    DO 30500 I=1,IVxNBOU
                        RVxBOUN(I,1) = RVxBOUN(I,1)/RPxDTOR
                        RVxBOUN(I,2) = RVxBOUN(I,2)/RPxDTOR
                        RVxBOUN(I,3) = RVxBOUN(I,3)/300.0D00
                        RVxBOUN(I,4) = RVxBOUN(I,4)*RPxREARTH
       30500        CONTINUE
                    RETURN
                ENDIF
                GO TO 20106
                END
```

RAYSUB Local Symbols

Name	Class	Type	Size
RVXELEV	param		
RVXAZI.	param		
RTXP2	local	REAL*8	8
ITXK.	local	INTEGER*4	4
RTXP3	local	REAL*8	8
IFXGEN.	local	INTEGER*4	4
IVXJ.	local	INTEGER*4	4
RVXRTMP	local	REAL*8	8
RTXS4	local	REAL*8	8
RVXUDXDS.	local	REAL*8	8
I	local	INTEGER*4	4
RVXUDYDS.	local	REAL*8	8
J	local	INTEGER*4	4
RVXUDZDS.	local	REAL*8	8
IFXSFL.	local	INTEGER*4	4
IFXBUN.	local	INTEGER*4	4
RVXLA	local	REAL*8	8

RAYSUB Local Symbols

Name	Class	Type	Size
IFXPPF.	local	INTEGER*4	4
RVXLAT0	local	REAL*8	8
LVXEND.	local	LOGICAL*4	4
IVXIII.	local	INTEGER*4	4
IVXNBL.	local	INTEGER*4	4
RVXMU2.	local	REAL*8	8
RVXAZI0	local	REAL*8	8
RTXRT1.	local	REAL*8	8
RTXAINC	local	REAL*8	8
RVXDELR2.	local	REAL*8	8
RVXLON0	local	REAL*8	8
RVXGPLTOT	local	REAL*8	8
RVXLO	local	REAL*8	8
RVXELEV0.	local	REAL*8	8
RVXDET.	local	REAL*8	8
RVXDEV.	local	REAL*8	8
RVXBNDL	local	REAL*8	1056
RVXMU	local	REAL*8	8
RVXHZ	local	REAL*8	8
RVXDDSZ1.	local	REAL*8	8
RVXDELR	local	REAL*8	8
RVXPPI.	local	REAL*8	8
IFXN4	local	INTEGER*4	4
RVXDDXHB.	local	REAL*8	8
RVXDDYHB.	local	REAL*8	8
IVXNBOU	local	INTEGER*4	4
RVXPPL.	local	REAL*8	8
IVXSCTRI1	local	INTEGER*4	4
RVXFREQ	local	REAL*8	8
LVXHCUT	local	LOGICAL*4	4
IVXSCTRI2	local	INTEGER*4	4
RVXDELZ	local	REAL*8	8
RTXDZCALC	local	REAL*8	8
IVXSCTRI3	local	INTEGER*4	4
RTXTEMP	local	REAL*8	8
RVXRALIM.	local	REAL*8	8
RVXCOSEL.	local	REAL*8	8
RVXXGRAD.	local	REAL*8	8
RVXYGRAD.	local	REAL*8	8
RVXRMAX	local	REAL*8	8
RVXCSUM	local	REAL*8	8
RVXH0	local	REAL*8	8
LVXRCUT	local	LOGICAL*4	4
RVXDXDZ	local	REAL*8	8
RVXCOFMAT	local	REAL*8	32

RAYSUB Local Symbols

Name	Class	Type	Size
IFXCCAL	local	INTEGER*4	4
RVXDYDZ	local	REAL*8	8
RVXSINEL	local	REAL*8	8
RVXGPLINC	local	REAL*8	8
IFXEND	local	INTEGER*4	4
IVXSCSING	local	INTEGER*4	4
RTXP1	local	REAL*8	8
RVXH	SCPS1A	REAL*8	43200
RVXV1	IONO3	REAL*8	8
RVXV2	IONO3	REAL*8	8
RVXSEZGEC	MISC	REAL*8	72
RVXLOSA	LOSSES	REAL*8	8
RVXFNSB	IONO1	REAL*8	24
RVXHB	IONO1	REAL*8	24
RVXLOSR	LOSSES	REAL*8	8
RVXLOSX	LOSSES	REAL*8	8
RVXLOSG	LOSSES	REAL*8	8
RVXHBOT	OTHER	REAL*8	8
RVXANGLIM	GORP	REAL*8	8
RVXHCUT	OTHER	REAL*8	8
RVXFSQU	OTHER	REAL*8	8
RPXHTOP	PRAM	REAL*8	8
RVXHMIN	MISC	REAL*8	8
RVXRPI	OTHER	REAL*8	8
RVXCX	MORE	REAL*8	8
RVXXX	IONO3	REAL*8	8
RVXCY	MORE	REAL*8	8
RVXCZ	MORE	REAL*8	8
RVXHBND	IONO3	REAL*8	8
IVXPARAM	MAINDAT	INTEGER*4	24
IVXLAUN	MAINDAT	INTEGER*4	12
RVXXI	START	REAL*8	8
RVXYI	START	REAL*8	8
RVXZI	START	REAL*8	8
IVXSSNUM	LPARM	INTEGER*4	4
RVXF40	TEMP1	REAL*8	8
IVXTIME	LPARM	INTEGER*4	20
RVXH5	END	REAL*8	8
RVXF65	TEMP1	REAL*8	8
RVXK1	TEMP1	REAL*8	8
IFXGRND	LPARM	INTEGER*4	4
RVXHL	TEMP1	REAL*8	8
RVXXF	END	REAL*8	8
ICXNSCP	SCPS1	INTEGER*4	4
RVXYF	END	REAL*8	8

RAYSUB Local Symbols

Name	Class	Type	Size
IVXSX	MORE	INTEGER*4	4
RVXZF	END	REAL*8	8
IVXSY	MORE	INTEGER*4	4
IVXSZ	MORE	INTEGER*4	4
IFXCASE	IONO3	INTEGER*4	4
RVXGRID	MAINDAT	REAL*8	48
RVXALPHA.	IONO1	REAL*8	8
RVXIONPT.	MAINDAT	REAL*8	86400
RVXLAUN	MAINDAT	REAL*8	56
RVXBETA	IONO1	REAL*8	8
RVXOPTION	MAINDAT	REAL*8	56
RVXETA1	IONO1	REAL*8	8
RVXETA2	IONO1	REAL*8	8
RVXLATI	START	REAL*8	8
RVXDD6.	GORP	REAL*8	8
RVXLONI	START	REAL*8	8
RVXLAT1	OTHER	REAL*8	8
RVXBOUN	RESULTS	REAL*8	1320
RVXLON1	OTHER	REAL*8	8
RVXX7	MISC	REAL*8	8
RVXY7	MISC	REAL*8	8
RVXYEAR	LPARM	REAL*8	8
RVXZ7	MISC	REAL*8	8
RVXA100	LPARM	REAL*8	8
RVXR1	MISC	REAL*8	8
RVXTIM.	LPARM	REAL*8	8
RVXR2	MISC	REAL*8	8
RVXB5	IONO2	REAL*8	24
RPXPI	PRAM	REAL*8	8
RVXB6	IONO2	REAL*8	24
RPXDTOR	PRAM	REAL*8	8
RVXA5	IONO2	REAL*8	24
RPXREARTH	PRAM	REAL*8	8
RVXLATSC.	SCPS1	REAL*8	14400
RVXA6	IONO2	REAL*8	24
RVXLONSC.	SCPS1	REAL*8	14400
RVXE5	IONO2	REAL*8	24
RVXE6	IONO2	REAL*8	24
RVXFNSQ	SCPS1A	REAL*8	43200

Line# Source Line

```
      SUBROUTINE ROTSEZ
C
C-------------------------------------------------------------------
C
C  ROTSEZ -- SUBROUTINE TO ESTABLISH THE TRANSFORMATION
C        MATRIX FROM S.E.Z. TO G.E.C. COORDINATES
C
C        CALLED BY: RAYSUB
C
C-------------------------------------------------------------------
C
C        AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
C
C        DATE:            07/30/86
C
C        VERSION:         1.1
C
C-------------------------------------------------------------------
C
C        REVISED:         07/25/86 -- INITIAL REVISION.  TRANSLATED
C                         FROM TEKTRONIX BASIC TO VAX FORTRAN BY
C                         ERIC L. STROBEL.
C
C                         07/30/86 -- V1.1.  Change over to use of
C                         REAL*8 precision in the calculations.
C
C-------------------------------------------------------------------
C
C       USES:   RVCxLA           LAT OF START POINT
C               RVCxLO           LON OF START POINT
C
C        TO CALCULATE THE ELEMENTS OF THE ROTATION MATRIX.
C
C       RETURNS:         RVCxSEZTX(3,3)  THE ROTATION MATRIX
C
C-------------------------------------------------------------------
C
        REAL*8 RVCxLA, RVCxLO, RVCxSEZTX(3,3), RTLx1
        REAL*8 RTLx2, RTLx3, RTLx4, RTLx5, RTLx6, RTLx7, RTLx8
        REAL*8 RTLx9, RTLx10, RTLx11, RTLx12, RTLx13, RTLx14
C
        COMMON /MISC/ RVCxSEZTX,RTLx5,RTLx6,RTLx7,RTLx8,RTLx9,RTLx10
        COMMON /OTHER/ RVCxLA, RVCxLO,RTLx11,RTLx12,RTLx13,RTLx14
C
        RTLx1 = DSIN(RVCxLA)
        RTLx2 = DCOS(RVCxLA)
        RTLx3 = DSIN(RVCxLO)
        RTLx4 = DCOS(RVCxLO)
        RVCxSEZTX(1,1) = RTLx1 * RTLx4
        RVCxSEZTX(1,2) = -RTLx3
```

```
Line#  Source Line

             RVCxSEZTX(1,3) = RTLx2 * RTLx4
             RVCxSEZTX(2,1) = RTLx1 * RTLx3
             RVCxSEZTX(2,2) = RTLx4
             RVCxSEZTX(2,3) = RTLx2 * RTLx3
             RVCxSEZTX(3,1) = -RTLx2
             RVCxSEZTX(3,2) = 0.0D00
             RVCxSEZTX(3,3) = RTLx1
             RETURN
             END
```

ROTSEZ Local Symbols

Name	Class	Type	Size
RTLX1	local	REAL*8	8
RTLX2	local	REAL*8	8
RTLX3	local	REAL*8	8
RTLX4	local	REAL*8	8
RVCXLA.	OTHER	REAL*8	8
RVCXLO.	OTHER	REAL*8	8
RVCXSEZTX	MISC	REAL*8	72
RTLX5	MISC	REAL*8	8
RTLX6	MISC	REAL*8	8
RTLX7	MISC	REAL*8	8
RTLX8	MISC	REAL*8	8
RTLX9	MISC	REAL*8	8
RTLX10.	MISC	REAL*8	8
RTLX11.	OTHER	REAL*8	8
RTLX12.	OTHER	REAL*8	8
RTLX13.	OTHER	REAL*8	8
RTLX14.	OTHER	REAL*8	8

Line# Source Line

```
      SUBROUTINE LATLON3
C
C-----------------------------------------------------------------
C
C   LATLON3 -- SUBROUTINE TO OBTAIN LAT, LON OF THE INITIAL
C         POINT IN THE RAYPATH INCREMENT.
C
C         CALLED BY: RAYSUB
C
C-----------------------------------------------------------------
C
C         AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
C
C         DATE:            07/30/86
C
C         VERSION:         1.1
C
C-----------------------------------------------------------------
C
C         REVISED:         07/25/86 -- INITIAL REVISION.  TRANSLATED
C                          FROM TEKTRONIX BASIC TO VAX FORTRAN BY
C                          ERIC L. STROBEL.
C
C                          07/30/86 -- V1.1.  Change over to use of
C                          REAL*8 precision in the calculations.
C
C-----------------------------------------------------------------
C
C        USES:   RVCxX,Y,Z        INITIAL X, Y, Z FOR THE RAYPATH
C                                          INCREMENT
C                RVCxMAT(3,3)     SEZ-GEC TRANSFORMATION MATRIX
C                RPCxR            EARTH RADIUS
C
C         TO CALCULATE THE CORRESPONDING LAT, LON FOR THE GIVEN
C                POINT.
C
C        RETURNS:        RVCxLA,LO        LAT, LON FOR THE INITIAL
C                                            RAYPATH POINT.
C
C-----------------------------------------------------------------
C
        REAL*8 RVCx1, RVCx2, RVCx3, RVCx4, RVCx5, RVCx6
        REAL*8 RPCxPI, RPCxDTOR, RPCxR
        REAL*8 RVCxMAT(3,3), RVCxX, RVCxY, RVCxZ
        REAL*8 RVCxLA, RVCxLO, RTLx1, RTLx2
C
        COMMON /START/ RVCxX, RVCxY, RVCxZ, RVCxLA, RVCxLO
        COMMON /PRAM/ RPCxPI, RPCxDTOR, RPCxR, RTLx2
        COMMON /MISC/ RVCxMAT,RVCx1,RVCx2,RVCx3,RVCx4,RVCx5,RVCx6
C
```

```
Line#  Source Line

              RTLx1 = RPCxR + RVCxZ
              RVCx4 = DSQRT(RVCxX*RVCxX + RVCxY*RVCxY + RTLx1*RTLx1)
       C
       C      RVCx1, 2, & 3 represent the GEC coordinates.
       C
              RVCx1 = RVCxMAT(1,1)*RVCxX + RVCxMAT(1,2)*RVCxY + RVCxMAT(1,3)
           *RTLx1
              RVCx2 = RVCxMAT(2,1)*RVCxX + RVCxMAT(2,2)*RVCxY + RVCxMAT(2,3)
           *RTLx1
              RVCx3 = RVCxMAT(3,1)*RVCxX + RVCxMAT(3,2)*RVCxY + RVCxMAT(3,3)
           *RTLx1
       C
              RVCx5 = DSQRT( RVCx1*RVCx1 + RVCx2*RVCx2 )
              RVCxLA = DASIN( RVCx3 / RVCx4 )
              RVCxLO = DASIN( RVCx2 / RVCx5 )
              IF (RVCx1.GE.0.0D00) RETURN
              RVCxLO = -RVCxLO + INTSIGN(RVCx2)*RPCxPI
              IF (INTSIGN(RVCx2).EQ.0) RVCxLO = RPCxPI
              RETURN
              END
```

LATLON3 Local Symbols

Name	Class	Type	Size
RTLX1	local	REAL*8	8
RVCX1	MISC	REAL*8	8
RVCX2	MISC	REAL*8	8
RVCX3	MISC	REAL*8	8
RVCX4	MISC	REAL*8	8
RVCX5	MISC	REAL*8	8
RVCX6	MISC	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RPCXDTOR.	PRAM	REAL*8	8
RPCXR	PRAM	REAL*8	8
RVCXMAT	MISC	REAL*8	72
RVCXX	START	REAL*8	8
RVCXY	START	REAL*8	8
RVCXZ	START	REAL*8	8
RVCXLA.	START	REAL*8	8
RVCXLO.	START	REAL*8	8
RTLX2	PRAM	REAL*8	8

Global Symbols

Name	Class	Type	Size
ANRANG.	extern	***	***

Global Symbols

Name	Class	Type	Size
END	common	***	32
ENDPT	extern	***	***
GCDEV	extern	***	***
GORP.	common	***	16
INTSIGN	extern	INTEGER*4	***
IONO1	common	***	80
IONO2	common	***	144
IONO3	common	***	36
IONOPAR	extern	***	***
LATLON3	FSUBRT	***	***
LOSS.	extern	***	***
LOSSES.	common	***	32
LPARM	common	***	52
MAINDAT	common	***	86596
MISC.	common	***	120
MORE.	common	***	36
NEWCS	extern	***	***
OTHER	common	***	48
PHSPL	extern	***	***
PRAM.	common	***	32
RAYSUB.	FSUBRT	***	***
RESULTS	common	***	1320
ROTSEZ.	FSUBRT	***	***
SCPS1	common	***	28804
SCPS1A.	common	***	86400
START	common	***	40
TEMP1	common	***	32
TILTS	extern	***	***

Code size = 1dce (7630)
Data size = 00bf (191)

```
Line#  Source Line

          SUBROUTINE IONOPAR(IVSxJ,IFSxN4,IFSxG,IVSxSCS,IVSxSCT1,IVSxSCT2
      *,IVSxSCT3)
C
C-----------------------------------------------------------------
C
C   IONOPAR -- SUBROUTINE TO CALCULATE THE PROPERTIES OF THE
C          IONOSPHERE FOR A GIVEN POSITION.
C
C          CALLED BY: RAYSUB
C
C          CALLS: TRIANG, INBOX, INTERP, CASE1-5
C
C-----------------------------------------------------------------
C
C          AUTHORS:          MICHAEL H. REILLY & ERIC L. STROBEL
C
C          DATE:             03/18/88
C
C          VERSION:          3.1
C
C-----------------------------------------------------------------
C
C          REVISED:          07/25/86 -- INITIAL REVISION.  TRANSLATED
C                            FROM TEKTRONIX BASIC TO VAX FORTRAN BY
C                            ERIC L. STROBEL.
C
C                            07/30/86 -- V1.1.  Change over to use of
C                            REAL*8 precision in the calculations.
C
C                            10/10/86 -- V2.0.  Altered to fit the new
C                            status of the old RAYTRACE program as a
C                            subroutine.  Important change in sorting
C                            out which triangle a pt. is in:  With large
C                            arrays of points the old algorithm would
C                            have taken nearly forever (REALLY!).  Since
C                            the points are now being defined on a grid,
C                            the problem is vastly simpler.
C
C                            09/01/87 -- V3.0.  Now incorporates the
C                            RADAR-C ionosphere.  The interpolation
C                            routine has been modularized.  Estimates
C                            of the nearest boundary are now reported.
C
C                            03/18/88 -- V3.1.  Some corrections have been
C                            made so that the manipulation of grid indices
C                            works when the date line is crossed by the
C                            grid.
C
C-----------------------------------------------------------------
C
```

Line# Source Line

```
C        USES:   IVSxJ             J-TH IONOSPHERE LAYER
C                IFSxN4,G          FLAGS
C                RVCxX,Y,Z7        INTERMEDIATE COORD. VALUES FROM
C                                     LAT, LON COMPUTATION
C                RVCxR1            ANOTHER INTERMEDIATE VALUE
C                RVCxLAI,LOI       INITIAL PT'S. LAT, LON
C                RVCxH5            NEW HEIGHT
C                ICCxN             NUMBER OF S.C.P.'S
C                RPCxRE            RADIUS OF THE EARTH
C                RVCxFN(1800,3)    X
C                RVCxH(1800,3)     X
C                RVCxLA(1800)      X-DATA FOR THE S.C.P.'S
C                RVCxLO(1800)      X
C                IVCxSZ            SIGN FOR Z
C                RVCxFREQ          WAVE FREQ. SQUARED
C
C         TO CALCULATE THE VALUES OF THE IONOSPHERIC PARAMETERS
C                 USED TO UPDATE THE RAYPATH.
C
C         RETURNS:
C                RVCxFB(3)         FN^2 VALUES OF THE LAYERS
C                RVCxHB(3)         H & Y VALUES OF THE LAYERS
C                RVCxALPH          X
C                RVCxBETA          X
C                RVCxET1           X-COEF. VALUES FOR MU^2
C                RVCxET2           X
C                RVCxB5&6          ]
C                RVCxE5&6          ]
C                RVCxV1            ]-SOME INTERMEDIATE VALUES
C                RVCxV2            ]
C                IVSxSCS           VBL. SIGNIFYING USE OF SINGLE
C                                           S.C.P.
C                IVSxSCT1-3        VBLS. FOR THE TRIANGLE OF S.C.P.'S
C                                           USED
C
C-----------------------------------------------------------------
C
         INTEGER IVSxSCS, IVSxSCT1, IVSxSCT2, IVSxSCT3
         INTEGER IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
         INTEGER IVCxSX, IVCxSY, IVSxJ, IFCxSN
         INTEGER IFSxN4, IFSxG, IVCxSZ, ICCxN
         INTEGER IFLxOUT, IVLxIN, IVLxFIN, IVCxCASE
C
         REAL*8 RVxGRID(6), RPCxRE, RPCxPI, RPCxDR
         REAL*8 RTLxD1, RTLxD2, RTLxD5, RTLx1
         REAL*8 RVLxTR1, RVLxTR2, RTLxTH
         REAL*8 RTLxH, RTLxI3, RTLxI4, RTLxI5
         REAL*8 RVCxA5(3), RVCxA6(3), RVCxLAI, RVCxLOI
         REAL*8 RVCxFN(1800,3),RVCxH(1800,3),RVCxLA(1800),RVCxLO(1800)
         REAL*8 RVCxALPH, RVCxBETA, RVCxHBND, RVCxF40, RVCxF65
```

Line#   Source Line

```
         REAL*8 RVCxET1, RVCxET2, RVCxFB(3), RVCxHB(3), RVCxB5(3)
         REAL*8 RVCxB6(3), RVCxE5(3), RVCxE6(3)
         REAL*8 RVCxV1, RVCxV2, RVCxH5, RVCxFREQ
         REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVLxHTV, RVLxSLV
         REAL*8 RVCxXI, RVCxYI, RVCxZI, RVCxXF, RVCxYF, RVCxZF
         REAL*8 RVCxK1, RVCxHL, RVCxXX, RVCxXL, RVCxXU, RVCxA0
         REAL*8 RVCxHB1, RVCxHU, RVCxB0, RVCxH2, RVCxYS, RVCxSL1
         REAL*8 RVCxSL2, RVCxH1L, RVCxA1, RVCxB1, RVCxC1, RVCxH1P
         REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4, RVCxHT5
         REAL*8 RVLxLATS, RVLxLONS, RVLxLATE, RVLxLONE, RVLxLOI
         REAL*8 RVLxF98, RVLxK2, RVCxH2P, RTCxA, RTCxB, RTCxC
C
         COMMON /MAINDAT/ RVxGRID
         COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RTCxA
         COMMON /SCPS1/ ICCxN,RVCxLA,RVCxLO
         COMMON /SCPS1A/ RVCxFN,RVCxH
         COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxB,RTCxC
         COMMON /START/ RVCxXI, RVCxYI, RVCxZI, RVCxLAI, RVCxLOI
         COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RVCxH5
         COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
         COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
         COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
         COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
         COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
         COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
         COMMON /TEMP3/ RTLxI3, RTLxI4, RTLxI5, RTLxD5
         COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
         COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RVCxSL1,RVCxSL2,RVCxH1L
         COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
         COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
         COMMON /RAID/ IFCxSN
C
C----------------------------------------------------------------------
C
10000    RTLxH = RVCxH5
         RVCxHBND = 0.0D00
         RVCxLAI = RVCxLAI / RPCxDR
         RVCxLOI = RVCxLOI / RPCxDR
         RVLxLOI = RVCxLOI
         IF (ICCxN.GT.1) THEN
C
C -------------------------------------------------------------------
C          If the number of points is not 1, then there is a grid.  So,
C            calculate the grid boundaries, then call INBOX to determine
C            whether the present location is w/in the grid.  If not in
C            the grid, then do a spherically symmetric ionosphere based
C            upon the nearest grid point's parameters.  Within the grid,
C            call TRIANG to determine the three points that are to be
C            used in the interpolation.
C -------------------------------------------------------------------
```

```
Line#  Source Line

C
                 RVLxLATS = RVxGRID(3)
                 RVLxLONS = RVxGRID(4)
                 RVLxLATE = RVLxLATS + (RVxGRID(5)-1.0D00)*RVxGRID(1)
                 RVLxLONE = RVLxLONS + (RVxGRID(6)-1.0D00)*RVxGRID(2)
C
C           If the grid spans the date line, it's longitude values
C             will not be negative.  While this is good for purposes
C             of the interpolation, ray points with west longitudes
C             will spuriously be considered to be outside the grid.
C             Hence, the need for the temporary copy of the longitude.
C             If necessary, the next statement will bring the longitude
C             of the ray point into the same longitude system as the
C             grid.
C
                 IF (RVLxLOI.LT.RVLxLONS) RVLxLOI = RVLxLOI + 360.0
C
                 IFLxOUT = -1
                 CALL INBOX(RVCxLAI,RVLxLOI,RVLxLATS,RVLxLONS,RVLxLATE
     #,RVLxLONE,IFLxOUT)
C
C           IFLxOUT = 1 means that the current location is outside the
C             ionospheric specification grid, so only the nearest grid
C             point is used.
C
                 IF (IFLxOUT.EQ.1) THEN
                     IFCxN4 = 1
                     IF (RVCxLAI.LT.RVLxLATS) THEN
                         ITLxA = 1
                     ELSE IF (RVCxLAI.GT.RVLxLATE) THEN
                         ITLxA = IDNINT(RVxGRID(5))
                     ELSE
                         ITLxA =IDNINT(((RVCxLAI-RVLxLATS)/RVxGRID(1))+1.0
                     ENDIF
                     IF (RVLxLOI.LT.RVLxLONS) THEN
                         ITLxB = 1
                     ELSE IF (RVLxLOI.GT.RVLxLONE) THEN
                         ITLxB = IDNINT(RVxGRID(6))
                     ELSE
                         ITLxB =IDNINT(((RVLxLOI-RVLxLONS)/RVxGRID(2))+1.0
                     ENDIF
                     IVCxSCS = IDNINT((ITLxA - 1.0D00)*RVxGRID(6) + ITLxB
                 ELSE
                     CALL TRIANG
                     IFCxN4 = 2
                 ENDIF
            ELSE
C
C           These are the values to use if only one specification point
C             is given.
```

Line# Source Line

```
C
              IVCxSCS = 1
              IFCxN4 = 1
              IFLxOUT = 1
          ENDIF
C
C          Some initializations.
C
          IVSxSCS = IVCxSCS
          IVSxSCT1 = IVCxSCT1
          IVSxSCT2 = IVCxSCT2
          IVSxSCT3 = IVCxSCT3
          IFSxN4 = IFCxN4
          RVCxET1 = 0.0D00
          RVCxET2 = 0.0D00
          RVCxALPH = 0.0D00
          RVCxBETA = 0.0D00
          RVCxXX = 0.0D00
          RVCxLOI = RVCxLOI * RPCxDR
          RVLxLOI = RVLxLOI * RPCxDR
          RVCxLAI = RVCxLAI * RPCxDR
C
C         For calculation purposes here, move slightly off of the
C           boundary, if we're at one.
C
          IF (IFSxG.EQ.2.OR.IFSxG.GE.6) RTLxH =
       #RTLxH + IVCxSZ*1.0D-02
C
C -------------------------------------------------------------
C          Now the enumeration of possible height regions begins.
C            First, handle what is needed for the free-space
C            underlying the ionosphere (below 40 km).  Also, the lower
C            part of the D layer is exponential and spherically sym-
C            metric, so do that now too, since it involves no
C            interpolation.  For full details see the RADAR-C report
C            cited in the documentation.
C -------------------------------------------------------------
C
          IF (RTLxH.LE.40.0D00) THEN
              IF (IVCxSZ.GT.0) THEN
C
C -------------------------------------------------------------
C          HBND is the (approximate in some cases) height of the
C             next boundary to be encountered.
C          IVSxJ is sort of an integer counterpart to HBND.
C -------------------------------------------------------------
C
                  RVCxHBND = 40.0D00
                  IVSxJ = 1
              ELSE
```

```
Line#  Source Line

                    RVCxHBND = 0.0D00
                    IVSxJ = 0
                ENDIF
                RETURN
            ENDIF
C
C           The lower D region of RADAR-C
C
            IF (RTLxH.LE.65.0D00) THEN
                RVCxXX = RVCxF40 * DEXP(RVCxK1 * (RTLxH - 40.0D00))
                RVCxALPH = RVCxK1 * RVCxXX / RVCxFREQ
                RVCxBETA = -RVCxK1 * RVCxALPH / 2.0D00
                IF (IVCxSZ.GT.0) THEN
                    RVCxHBND = 65.0D00
                    IVSxJ = 2
                ELSE
                    RVCxHBND = 40.0D00
                    IVSxJ = 1
                ENDIF
                RETURN
            ENDIF
C
C ------------------------------------------------------------------
C            Here is where, if the case is a non-spherically symmetric
C              one, that interpolation begins to be needed.
C
C            Note:   FB(1)   -->   foE ** 2
C                    FB(2)   -->   foF1 ** 2
C                    FB(3)   -->   foF2 ** 2
C
C                    HB(1)   -->   hmF1
C                    HB(2)   -->   hmF2
C                    HB(3)   -->   YmF2
C
C            The interpolation is done as follows,
C               X = A + B*Lat + E*Lon, where A, B, and E are determined
C               by setting up the three equations at the three grid
C               points surrounding the point in question.  This is done
C               in INTERP.
C ------------------------------------------------------------------
C
C           Recall that IFSxN4 = 1 for the spherically symmetric case.
C
C ------------------------------------------------------------------
C
            IF (IFSxN4.EQ.1) THEN
                RVCxFB(1) = RVCxFN(IVSxSCS,1)
            ELSE
                RVCxV1 = RPCxRE + RTLxH
                RVCxV2 = RVCxV1 * DCOS(RVCxLAI)
```

Line# Source Line

```
            RTLxI3 = RVCxLA(IVSxSCT1)*RVCxLO(IVSxSCT2)-RVCxLA(IVSxSCT
     #*RVCxLO(IVSxSCT1)
            RTLxI4 = RVCxLA(IVSxSCT2)*RVCxLO(IVSxSCT3)-RVCxLA(IVSxSCT
     #*RVCxLO(IVSxSCT2)
            RTLxI5 = RVCxLA(IVSxSCT3)*RVCxLO(IVSxSCT1)-RVCxLA(IVSxSCT
     #*RVCxLO(IVSxSCT3)
            RTLxD5 = RTLxI3 + RTLxI4 + RTLxI5
C
C        IVLxIN and FIN are used to get INTERP to operate only on wha
C           is needed to do the calculations for the D & E regions.
C
            IVLxIN = 1
            IVLxFIN = 1
            CALL INTERP(IVLxIN, IVLxFIN)
            RVCxFB(1) = RVCxA5(1)+RVCxB5(1)*RVCxLAI+RVCxE5(1)*RVLxLOI
         ENDIF
         IF (RTLxH.LE.98.0D00) THEN
C
C        The upper D layer is exponential.
C
            RVLxF98 = 0.64D00 * RVCxFB(1)
            RVLxK2 = (DLOG(RVLxF98)-DLOG(RVCxF65))/33.0D00
            RVCxXX = RVCxF65 * DEXP(RVLxK2 * (RTLxH - 65.0D00))
            RVCxALPH = RVLxK2 * RVCxXX / RVCxFREQ
            RVCxBETA = -RVLxK2 * RVCxALPH / 2.0D00
            IF (IVCxSZ.GT.0) THEN
                RVCxHBND = 98.0D00
                IVSxJ = 3
            ELSE
                RVCxHBND = 65.0D00
                IVSxJ = 2
            ENDIF
            IF (IFSxN4.NE.1) THEN
                RTLx1 = ((RTLxH - 65.0D00) * RVCxXX)/
     #(33.0D00 * RVCxFB(1) * RVCxFREQ)
                RVCxET1 = -RTLx1 * RVCxB5(1) / RVCxV1
                RVCxET2 = RTLx1 * RVCxE5(1) / RVCxV2
            ENDIF
            RETURN
         ENDIF
         IF (RTLxH.LE.RVCxHL) THEN
C
C        The E layer is parabolic, with only foE as a non-constant
C           parameter.
C
            RVLxHTV = (RTLxH - 110.0D00) / 20.0D00
            RVLxSLV = RVCxFB(1) /20.0D00
            RVCxXX = RVCxFB(1) * (1.0D00 - RVLxHTV*RVLxHTV)
            RVCxALPH = -2.0D00 * RVLxHTV * RVLxSLV / RVCxFREQ
            RVCxBETA = RVLxSLV / (20.0D00 * RVCxFREQ)
```

Line# Source Line

```
            IF (IVCxSZ.GT.0) THEN
                RVCxHBND = RVCxHL
                IVSxJ = 4
            ELSE
                RVCxHBND = 98.0D00
                IVSxJ = 3
            ENDIF
            IF (IFSxN4.NE.1) THEN
                RTLx1 = (1.0D00 - RVLxHTV*RVLxHTV) / RVCxFREQ
                RVCxET1 = -RTLx1 * RVCxB5(1) / RVCxV1
                RVCxET2 = RTLx1 * RVCxE5(1) / RVCxV2
            ENDIF
            RETURN
        ENDIF
C
C        Now, get the rest of the values needed to do the
C          calculations in the F1 & F2 layers.
C
        IF (IFSxN4.EQ.1) THEN
            RVCxFB(2) = RVCxFN(IVSxSCS,2)
            RVCxFB(3) = RVCxFN(IVSxSCS,3)
            RVCxHB(1) = RVCxH(IVSxSCS,1)
            RVCxHB(2) = RVCxH(IVSxSCS,2)
            RVCxHB(3) = RVCxH(IVSxSCS,3)
        ELSE
            IVLxIN = 2
            IVLxFIN = 3
            CALL INTERP(IVLxIN, IVLxFIN)
C
C ------------------------------------------------------------------
C        The F1 values may be 0 in this model if the F1 isn't
C          seen in the profile. This botches up the interpolation
C          by introducing an artificially large gradient.  This
C          gets fixed by the next snippet of code.
C ------------------------------------------------------------------
C
            IF (RVCxH(IVSxSCT1,1).EQ.0.0.OR.RVCxH(IVSxSCT2,1).EQ.
     #0.0.OR.RVCxH(IVSxSCT3,1).EQ.0.0) THEN
                RVCxA5(2) = 0.0
                RVCxB5(2) = 0.0
                RVCxE5(2) = 0.0
                RVCxA6(1) = 0.0
                RVCxB6(1) = 0.0
                RVCxE6(1) = 0.0
            ENDIF
C
            RVCxFB(2) = RVCxA5(2) + RVCxB5(2)*RVCxLAI
     #+ RVCxE5(2)*RVLxLOI
            RVCxFB(3) = RVCxA5(3) + RVCxB5(3)*RVCxLAI
     #+ RVCxE5(3)*RVLxLOI
```

Line# Source Line

```
                RVCxHB(1) = RVCxA6(1) + RVCxB6(1)*RVCxLAI
         #+ RVCxE6(1)*RVLxLOI
                RVCxHB(2) = RVCxA6(2) + RVCxB6(2)*RVCxLAI
         #+ RVCxE6(2)*RVLxLOI
                RVCxHB(3) = RVCxA6(3) + RVCxB6(3)*RVCxLAI
         #+ RVCxE6(3)*RVLxLOI
            ENDIF
C
C ----------------------------------------------------------------------
C          From here on, the logic is complicated, but it follows the
C            procedures in the RADAR-C report.  Basically, it must be
C            determined whether the F1 is linear or parabolic, and
C            where the E, valley, F1, F2, and topside profiles fall
C            and where their intersection points are.
C ----------------------------------------------------------------------
C
            RVCxXL = 0.8516D00 * 0.8516D00 * RVCxFB(1)
            RVCxXU = 0.98D00 * 0.98D00 * RVCxFB(1)
            RVCxHU = RVCxHB(2) - RVCxHB(3) * DSQRT(1.0D00 -
         #(RVCxXU/RVCxFB(3)))
            RVCxHB1 = 0.75D00 * RVCxHB(1)
            RVCxA0 = (RVCxHU*RVCxXL - RVCxHL*RVCxXU)/(RVCxHU - RVCxHL)
            RVCxB0 = (RVCxXU - RVCxXL) / (RVCxHU - RVCxHL)
            RVCxHT5 = RVCxHB(2) + 0.25D00*RVCxHB(3)
            IF (RVCxFB(2).LE.RVCxXU) THEN
                IVCxCASE = 1
                GO TO 20000
            ENDIF
            IF (RVCxFB(2).GT.RVCxFB(3)) GO TO 10750
            RVCxH2 = RVCxHB(2) - RVCxHB(3) * DSQRT(1.0D00 -
         #(RVCxFB(2)/RVCxFB(3)))
            IF (RVCxHB(1).LE.RVCxH2) GO TO 10750
            RVCxYS = MAX(1.0D00, RVCxH2 - RVCxHB1)
            RVCxSL1 = RVCxFB(2) / RVCxYS
            RVCxSL2 = 2.0D00 * RVCxFB(3) * (RVCxHB(2) - RVCxH2)/
         #(RVCxHB(3)*RVCxHB(3))
            IF (RVCxSL1.GE.RVCxSL2) THEN
                RVCxH1L = (RVCxHB1*RVCxSL1 + RVCxA0)/
         #(RVCxSL1 - RVCxB0)
                IF (RVCxH1L.LT.RVCxHU) THEN
                    IVCxCASE = 3
                ELSE
                    IVCxCASE = 1
                ENDIF
                GO TO 20000
            ENDIF
10750       RVCxA1 = 16.0D00 * RVCxFB(2) / (RVCxHB(1)*RVCxHB(1))
            RVCxB1 = (32.0D00 * RVCxFB(2) / RVCxHB(1)) - RVCxB0
            RVCxC1 = 15.0D00 * RVCxFB(2) + RVCxA0
            RTLxD1 = RVCxB1*RVCxB1 - 4.0D00 * RVCxA1 * RVCxC1
```

Line# Source Line

```
            IF (RTLxD1.LT.0.0D00) THEN
                IVCxCASE = 1
                GO TO 20000
            ENDIF
            RVCxH1P = (RVCxB1 - INTSIGN(RVCxB1) * DSQRT(RTLxD1))
         #/(2.0D00 * RVCxA1)
            RVCxA2 = (RVCxFB(3)/(RVCxHB(3)*RVCxHB(3))) -
         #(16.0D00 * RVCxFB(2)/(RVCxHB(1)*RVCxHB(1)))
            RVCxB2 = (32.0D00 * RVCxFB(2)/RVCxHB(1)) -
         #(2.0D00 * RVCxFB(3)  * RVCxHB(2)/(RVCxHB(3)*RVCxHB(3)))
            RVCxC2 = RVCxFB(3)*(((RVCxHB(2)*RVCxHB(2))/(RVCxHB(3)*
         #RVCxHB(3))) - 1.0D00) - 15.0D00 * RVCxFB(2)
            RTLxD2 = RVCxB2 * RVCxB2 - 4.0D00 * RVCxA2 * RVCxC2
            IF (RTLxD2.LT.0.0D00) THEN
                IVCxCASE = 1
                GO TO 20000
            ENDIF
            RVCxHT3 = (-RVCxB2 + INTSIGN(RVCxB2) * DSQRT(RTLxD2))/
         #(2.0D00 * RVCxA2)
            RVCxHT4 = (-RVCxB2 - INTSIGN(RVCxB2)*DSQRT(RTLxD2))/
         #(2.0D00 * RVCxA2)
            IF (RVCxH1P.LE.RVCxHU) THEN
                RVCxH2P = (RVCxB1 + INTSIGN(RVCxB1)*DSQRT(RTLxD1))/
         #(2.0D00 * RVCxA1)
                IF (RVCxH2P.LT.RVCxHU) THEN
                    IVCxCASE = 5
                    GO TO 20000
                ENDIF
                IVCxCASE = 2
                IF (RVCxB2.GT.0.0D00) THEN
                    RVLxTR1 = RVCxHT4
                    RVLxTR2 = RVCxHT3
                    IFCxSN = 1
                ELSE
                    RVLxTR1 = RVCxHT3
                    RVLxTR2 = RVCxHT4
                    IFCxSN = -1
                ENDIF
                RTLxTH = RVLxTR1
10850           IF (RTLxTH.GT.RVCxHU.AND.RTLxTH.LT.RVCxHT5) THEN
                    RVCxHT4 = RTLxTH
                    GO TO 20000
                ENDIF
                IF (RTLxTH.EQ.RVLxTR2) THEN
C
C             Some diagnostics that are of the nature of, "The program
C               can't reach this point, but..."
C
                    PRINT *, ' EXCEEDINGLY WIERD CASE: POINT #1'
                    PRINT *, ' HT3:',RVCxHT3,'   HT4:',RVCxHT4
```

```
Line#   Source Line

                      PRINT *, ' H5 :',RVCxH5,'    B2 :',RVCxB2
                      PRINT *, ' LAT:',RVCxLAI/RPCxDR,'    LON:',
     #RVCxLOI/RPCxDR
                      PRINT *, ' HT5:',RVCxHT5,'    HU :',RVCxHU
                      PRINT *, ' H1P:',RVCxH1P
                      STOP
                  ELSE
                      RTLxTH = RVLxTR2
                      IFCxSN = -IFCxSN
                      GO TO 10850
                  ENDIF
              ENDIF
              IF (RVCxHT3.LE.RVCxH1P) THEN
                  IVCxCASE = 1
                  GO TO 20000
              ENDIF
              IF (RVCxHT3.LT.RVCxHT4.AND.RVCxHT4.LT.RVCxHT5) THEN
                  IVCxCASE = 4
              ELSE
C
C        Another diagnostic as outlined above.
C
                  PRINT *, ' EXCEEDINGLY WIERD CASE: POINT #2'
                  STOP
              ENDIF
C
C        These routines continue the calculations, now that
C           it has been decided which path to take.
C
20000         IF (IVCxCASE.EQ.1) THEN
                  CALL CASE1(RTLxH, IVSxJ)
                  RETURN
              ELSE IF (IVCxCASE.EQ.2) THEN
                  CALL CASE2(RTLxH, IVSxJ)
                  RETURN
              ELSE IF (IVCxCASE.EQ.3) THEN
                  CALL CASE3(RTLxH, IVSxJ)
                  RETURN
              ELSE IF (IVCxCASE.EQ.4) THEN
                  CALL CASE4(RTLxH, IVSxJ)
              ELSE
                  CALL CASE5(RTLxH, IVSxJ)
              ENDIF
              RETURN
              END
```

IONOPAR Local Symbols

Name	Class	Type	Size
IVSXSCT3.	param		
IVSXSCT2.	param		
IVSXSCT1.	param		
IVSXSCS	param		
IFSXG	param		
IFSXN4.	param		
IVSXJ	param		
RVLXLATS.	local	REAL*8	8
RTLXD2.	local	REAL*8	8
RVLXSLV	local	REAL*8	8
ITLXA	local	INTEGER*4	4
ITLXB	local	INTEGER*4	4
RVLXF98	local	REAL*8	8
RVLXLONS.	local	REAL*8	8
RVLXK2.	local	REAL*8	8
RTLXH	local	REAL*8	8
IVLXIN.	local	INTEGER*4	4
IVLXFIN	local	INTEGER*4	4
RVLXTR1	local	REAL*8	8
RVLXTR2	local	REAL*8	8
RTLXTH.	local	REAL*8	8
IFLXOUT	local	INTEGER*4	4
RVLXLOI	local	REAL*8	8
RVLXLATE.	local	REAL*8	8
RVLXLONE.	local	REAL*8	8
RTLX1	local	REAL*8	8
RVLXHTV	local	REAL*8	8
RTLXD1.	local	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXFB.	IONO1	REAL*8	24
RVCXHB.	IONO1	REAL*8	24
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXH5.	END	REAL*8	8
RVCXFREQ.	OTHER	REAL*8	8
RVCXL1.	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXXI.	START	REAL*8	8
RVCXYI.	START	REAL*8	8
RVCXZI.	START	REAL*8	8
RVCXXF.	END	REAL*8	8

IONOPAR Local Symbols

Name	Class	Type	Size
RVCXYF.	END	REAL*8	8
IVCXSCS	TEMP2	INTEGER*4	4
RVCXZF.	END	REAL*8	8
RVCXK1.	TEMP1	REAL*8	8
IVCXSCT1.	TEMP2	INTEGER*4	4
RVCXHL.	TEMP1	REAL*8	8
IVCXSCT2.	TEMP2	INTEGER*4	4
RVCXXX.	IONO3	REAL*8	8
IVCXSCT3.	TEMP2	INTEGER*4	4
IFCXN4.	TEMP2	INTEGER*4	4
RVCXXL.	VAR1	REAL*8	8
RVCXXU.	VAR1	REAL*8	8
IVCXSX.	MORE	INTEGER*4	4
RVCXA0.	VAR1	REAL*8	8
IVCXSY.	MORE	INTEGER*4	4
RVCXHB1	VAR2	REAL*8	8
IFCXSN.	RAID	INTEGER*4	4
IVCXSZ.	MORE	INTEGER*4	4
RVCXHU.	VAR1	REAL*8	8
ICCXN	SCPS1	INTEGER*4	4
RVCXB0.	VAR1	REAL*8	8
RVCXH2.	VAR2	REAL*8	8
RVCXYS.	VAR2	REAL*8	8
RVCXSL1	VAR2	REAL*8	8
IVCXCASE.	IONO3	INTEGER*4	4
RVCXSL2	VAR2	REAL*8	8
RVXGRID	MAINDAT	REAL*8	48
RVCXH1L	VAR2	REAL*8	8
RVCXA1.	VAR3	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RVCXB1.	VAR3	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RVCXC1.	VAR3	REAL*8	8
RPCXDR.	PRAM	REAL*8	8
RVCXH1P	VAR3	REAL*8	8
RVCXA2.	VAR4	REAL*8	8
RVCXB2.	VAR4	REAL*8	8
RTLXD5.	TEMP3	REAL*8	8
RVCXC2.	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RTLXI3.	TEMP3	REAL*8	8
RTLXI4.	TEMP3	REAL*8	8
RTLXI5.	TEMP3	REAL*8	8

IONOPAR Local Symbols

Name	Class	Type	Size
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXLAI	START	REAL*8	8
RVCXLOI	START	REAL*8	8
RVCXH2P	VAR3	REAL*8	8
RVCXFN.	SCPS1A	REAL*8	43200
RTCXA	PRAM	REAL*8	8
RVCXH	SCPS1A	REAL*8	43200
RTCXB	OTHER	REAL*8	8
RVCXLA.	SCPS1	REAL*8	14400
RTCXC	OTHER	REAL*8	8
RVCXLO.	SCPS1	REAL*8	14400
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXHBND.	IONO3	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXET1	IONO1	REAL*8	8

Line# Source Line

```
      SUBROUTINE INTERP(IVSxIN, IVSxFIN)
C
C---------------------------------------------------------------
C
C   INTERP -- SUBROUTINE TO PERFORM THE LATITUDE AND LONGITUDE
C        INTERPOLATION REQUIRED BY IONOPAR
C
C        CALLED BY: IONOPAR
C
C---------------------------------------------------------------
C
C        AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
C
C        DATE:            09/01/87
C
C        VERSION:         1.0
C
C---------------------------------------------------------------
C
C        REVISED:         09/01/87 -- V1.0.  Initial revision.
C
C---------------------------------------------------------------
C
C        USES:   IVSxIN & IVSxFIN        TO DETERMINE WHICH SETS
C                         OF INTERPOLATION COEFFICIENTS TO DO.
C
C         The interpolation is done as follows,
C            X = A + B*Lat + E*Lon, where A, B, and E are determined
C            by setting up the three equations at the three grid
C            points surrounding the point in question.  These
C            equations are then solved by use of Kramer's rule.
C
C ---------------------------------------------------------------
C
C        RETURNS:       /IONO2/      THE COMMON BLOCK CONTAINING THE
C                         INTERPOLATION COEFFICIENTS (RVCx[A,B,E](5,6))
C                         WHERE A, B, AND E ARE AS ABOVE, AND THE 5
C                         REPRESENTS A FREQUENCY COEFFICIENT AND THE 6
C                         REPRESENTS A HEIGHT COEFFICIENT.
C
C ---------------------------------------------------------------
C
      INTEGER IVSxIN, IVSxFIN, IVCxSCT1, IVCxSCT2
      INTEGER IVCxSCT3, ICCxN, IVCxSCS, ITLx1
C
      REAL*8 RVCxI3, RVCxI4, RVCxI5, RVCxD5
      REAL*8 RVCxFN(1800,3), RVCxH(1800,3), RVCxLA(1800)
      REAL*8 RVCxLO(1800), RVCxA5(3), RVCxA6(3), RVCxB5(3)
      REAL*8 RVCxB6(3), RVCxE5(3), RVCxE6(3)
C
```

```
Line#  Source Line

               COMMON /SCPS1/ ICCxN,RVCxLA,RVCxLO
               COMMON /SCPS1A/ RVCxFN,RVCxH
               COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
               COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, ITLx1
               COMMON /TEMP3/ RVCxI3, RVCxI4, RVCxI5, RVCxD5
      C
               DO 10000 I = IVSxIN, IVSxFIN
                  RVCxA5(I) = (RVCxFN(IVCxSCT1,I)*RVCxI4 + RVCxFN(IVCxSCT2,I)
             #*RVCxI5 + RVCxFN(IVCxSCT3,I)*RVCxI3)/RVCxD5
                  RVCxA6(I) = (RVCxH(IVCxSCT1,I)*RVCxI4 + RVCxH(IVCxSCT2,I)
             #*RVCxI5 + RVCxH(IVCxSCT3,I)*RVCxI3)/RVCxD5
                  RVCxB5(I) = RVCxFN(IVCxSCT1,I)*(RVCxLO(IVCxSCT2)-RVCxLO(
             #IVCxSCT3)) + RVCxFN(IVCxSCT2,I)*(RVCxLO(IVCxSCT3)-
             #RVCxLO(IVCxSCT1))
                  RVCxB5(I) = (RVCxB5(I) + RVCxFN(IVCxSCT3,I)*(RVCxLO(
             #IVCxSCT1)-RVCxLO(IVCxSCT2)))/RVCxD5
                  RVCxB6(I) = RVCxH(IVCxSCT1,I)*(RVCxLO(IVCxSCT2)-RVCxLO(
             #IVCxSCT3)) + RVCxH(IVCxSCT2,I)*(RVCxLO(IVCxSCT3)-
             #RVCxLO(IVCxSCT1))
                  RVCxB6(I) = (RVCxB6(I) + RVCxH(IVCxSCT3,I)*(RVCxLO(
             #IVCxSCT1)-RVCxLO(IVCxSCT2)))/RVCxD5
                  RVCxE5(I) = RVCxFN(IVCxSCT1,I)*(RVCxLA(IVCxSCT3)-RVCxLA(
             #IVCxSCT2)) + RVCxFN(IVCxSCT2,I)*(RVCxLA(IVCxSCT1)-
             #RVCxLA(IVCxSCT3))
                  RVCxE5(I) = (RVCxE5(I) + RVCxFN(IVCxSCT3,I)*(RVCxLA(
             #IVCxSCT2)-RVCxLA(IVCxSCT1)))/RVCxD5
                  RVCxE6(I) = RVCxH(IVCxSCT1,I)*(RVCxLA(IVCxSCT3)-RVCxLA
             #IVCxSCT2)) + RVCxH(IVCxSCT2,I)*(RVCxLA(IVCxSCT1)-
             #RVCxLA(IVCxSCT3))
                  RVCxE6(I) = (RVCxE6(I) + RVCxH(IVCxSCT3,I)*(RVCxLA(
             #IVCxSCT2)-RVCxLA(IVCxSCT1)))/RVCxD5
      10000       CONTINUE
               RETURN
               END
```

INTERP Local Symbols

Name	Class	Type	Size
IVSXFIN	param		
IVSXIN.	param		
I	local	INTEGER*4	4
IVCXSCT1.	TEMP2	INTEGER*4	4
IVCXSCT2.	TEMP2	INTEGER*4	4
IVCXSCT3.	TEMP2	INTEGER*4	4
ICCXN	SCPS1	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
ITLX1	TEMP2	INTEGER*4	4
RVCXI3.	TEMP3	REAL*8	8

INTERP Local Symbols

Name	Class	Type	Size
RVCXI4.	TEMP3	REAL*8	8
RVCXI5.	TEMP3	REAL*8	8
RVCXD5.	TEMP3	REAL*8	8
RVCXFN.	SCPS1A	REAL*8	43200
RVCXH	SCPS1A	REAL*8	43200
RVCXLA.	SCPS1	REAL*8	14400
RVCXLO.	SCPS1	REAL*8	14400
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24

Line# Source Line

```
      SUBROUTINE CASE1(RVSxH, IVSxJ)
C
C-----------------------------------------------------------------
C
C  CASE1 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
C       IS IN THE CASE 1 PROFILE
C
C       CALLED BY: IONOPAR
C
C       CALLS:  PGVAL, PGF2, PGFB
C
C-----------------------------------------------------------------
C
C       AUTHOR:          ERIC L. STROBEL & MICHAEL H. REILLY
C
C       DATE:            09/01/87
C
C       VERSION:         1.0
C
C-----------------------------------------------------------------
C
C       REVISED:         09/01/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C       USES:         RVSxH          The current height.
C
C          To determine which part of the CASE 1 profile is being
C        operated on.  The CASE 1 profile consists of (above the E)
C        the linear valley, the parabolic F2, and the topside.
C
C       RETURNS:      IVSxJ          This helps distinguish which
C                                boundary is being approached.
C
C-----------------------------------------------------------------
C
      INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY
C
      REAL*8 RVSxH, RVCxHU, RVCxHBND, RVCxHL, RVCxHT5
      REAL*8 RPCxHTP, RPCxPI, RPCxDR, RPCxRE
      REAL*8 RVCxV1, RVCxV2, RVCxXX
      REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU
      REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4
      REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RVCxRPI
      REAL*8 RVCxHCT, RTLx1, RTLx2
C
      COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
      COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
      COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
      COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
```

Line# Source Line

```
          COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RTLx1, RTLx2
          COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
          COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
C
          IF (RVSxH.LE.RVCxHU) THEN
C
C         We're in the valley region, so go calculate the parameters
C           from the valley profile.
C
              CALL PGVAL(RVSxH)
              IF (IVCxSZ.GT.0) THEN
                  RVCxHBND = RVCxHU
                  IVSxJ = 5
              ELSE
                  RVCxHBND = RVCxHL
                  IVSxJ = 4
              ENDIF
              RETURN
          ENDIF
          IF (RVSxH.LE.RVCxHT5) THEN
C
C         We're in the F2 region, so go calculate the parameters
C           from the F2 profile.
C
              CALL PGF2(RVSxH)
              IF (IVCxSZ.GT.0) THEN
                  RVCxHBND = RVCxHT5
                  IVSxJ = 6
              ELSE
                  RVCxHBND = RVCxHU
                  IVSxJ = 5
              ENDIF
              RETURN
          ENDIF
          IF (RVSxH.LE.RPCxHTP) THEN
C
C         We're in the topside region, so go calculate the parameters
C           from the topside profile.
C
              CALL PGFB(RVSxH)
              IF (IVCxSZ.GT.0) THEN
                  RVCxHBND = RPCxHTP
                  IVSxJ = 7
              ELSE
                  RVCxHBND = RVCxHT5
                  IVSxJ = 6
              ENDIF
          ELSE
C
C         We're beyond the cutoff of the ionosphere.
```

```
Line#  Source Line

       C
                  IF (IVCxSZ.GT.0) THEN
                     RVCxHBND = RVCxHCT
                  ELSE
                     RVCxHBND = RPCxHTP
                  ENDIF
               ENDIF
       C
       C       Take care of things if the ray is headed toward a
       C          boundary that's above the cutoff height.
       C
               IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
                  RVCxHBND = RVCxHCT
                  IVSxJ = 1
               ENDIF
       C
               RETURN
               END
```

CASE1 Local Symbols

Name	Class	Type	Size
IVSXJ	param		
RVSXH	param		
RVCXXX	IONO3	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXK1	TEMP1	REAL*8	8
RVCXXL	VAR1	REAL*8	8
RVCXXU	VAR1	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXFREQ	OTHER	REAL*8	8
RVCXRPI	OTHER	REAL*8	8
RVCXHCT	OTHER	REAL*8	8
RTLX1	VAR1	REAL*8	8
RTLX2	VAR1	REAL*8	8
IVCXSZ	MORE	INTEGER*4	4
IVCXCASE	IONO3	INTEGER*4	4
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
RVCXHU	VAR1	REAL*8	8

CASE1 Local Symbols

Name	Class	Type	Size
RVCXHBND.	IONO3	REAL*8	8
RVCXHL.	TEMP1	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RPCXDR.	PRAM	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8

Line# Source Line

```
      SUBROUTINE CASE2(RVSxH, IVSxJ)
C
C-----------------------------------------------------------------
C
C   CASE2 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
C        IS IN THE CASE 2 PROFILE
C
C        CALLED BY: IONOPAR
C
C        CALLS:  PGVAL, PGF1P, PGF2, PGFB
C
C-----------------------------------------------------------------
C
C        AUTHOR:          ERIC L. STROBEL & MICHAEL H. REILLY
C
C        DATE:            09/01/87
C
C        VERSION:         1.0
C
C-----------------------------------------------------------------
C
C        REVISED:         09/01/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C        USES:        RVSxH          The current height.
C
C            To determine which part of the CASE 2 profile is being
C         operated on.  The CASE 2 profile consists of (above the E)
C         the linear valley, a parabolic F1, the parabolic F2,
C         and the topside.
C
C        RETURNS:     IVSxJ          This helps distinguish which
C                                  boundary is being approached.
C
C-----------------------------------------------------------------
C
      INTEGER IVSxJ, IVCxSZ, IVCxSX, IVCxSY, IVCxCASE
C
      REAL*8 RVSxH, RVCxH1P, RVCxHBND, RVCxHL, RVCxHT4
      REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR, RPCxRE
      REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RVCxRPI
      REAL*8 RVCxV1, RVCxV2, RVCxXX
      REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxA1, RVCxB1, RVCxC1
      REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RTCxA
C
      COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
      COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
      COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
      COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
```

Line# Source Line

```
      COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
      COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RTCxA
      COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
C
      IF (RVSxH.LE.RVCxH1P) THEN
C
C     We're in the valley region, so go calculate the parameters
C       from the valley profile.
C
          CALL PGVAL(RVSxH)
          IF (IVCxSZ.GT.0) THEN
              RVCxHBND = RVCxH1P
              IVSxJ = 5
          ELSE
              RVCxHBND = RVCxHL
              IVSxJ = 4
          ENDIF
          RETURN
      ENDIF
      IF (RVSxH.LE.RVCxHT4) THEN
C
C     We're in the F1 region, so go calculate the parameters
C       from the F1 profile.
C
          CALL PGF1P(RVSxH)
          IF (IVCxSZ.GT.0) THEN
              RVCxHBND = RVCxHT4
              IVSxJ = 6
          ELSE
              RVCxHBND = RVCxH1P
              IVSxJ = 5
          ENDIF
          RETURN
      ENDIF
      IF (RVSxH.LE.RVCxHT5) THEN
C
C     We're in the F2 region, so go calculate the parameters
C       from the F2 profile.
C
          CALL PGF2(RVSxH)
          IF (IVCxSZ.GT.0) THEN
              RVCxHBND = RVCxHT5
              IVSxJ = 7
          ELSE
              RVCxHBND = RVCxHT4
              IVSxJ = 6
          ENDIF
          RETURN
      ENDIF
      IF (RVSxH.LE.RPCxHTP) THEN
```

Line# Source Line

```
C
C       We're in the topside region, so go calculate the parameters
C         from the topside profile.
C
            CALL PGFB(RVSxH)
            IF (IVCxSZ.GT.0) THEN
                RVCxHBND = RPCxHTP
                IVSxJ = 8
            ELSE
                RVCxHBND = RVCxHT5
                IVSxJ = 7
            ENDIF
        ELSE
C
C       We're beyond the cutoff of the ionosphere.
C
            IF (IVCxSZ.GT.0) THEN
                RVCxHBND = RVCxHCT
            ELSE
                RVCxHBND = RPCxHTP
            ENDIF
        ENDIF
C
C       Take care of things if the ray is headed toward a
C          boundary that's above the cutoff height.
C
        IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
            RVCxHBND = RVCxHCT
            IVSxJ = 1
        ENDIF
C
        RETURN
        END
```

CASE2 Local Symbols

Name	Class	Type	Size
IVSXJ	param		
RVSXH	param		
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXFREQ.	OTHER	REAL*8	8
RVCXRPI	OTHER	REAL*8	8
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXXX.	IONO3	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8

CASE2 Local Symbols

Name	Class	Type	Size
RVCXK1	TEMP1	REAL*8	8
RVCXA1	VAR3	REAL*8	8
RVCXB1	VAR3	REAL*8	8
RVCXC1	VAR3	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RTCXA	VAR3	REAL*8	8
IVCXSZ	MORE	INTEGER*4	4
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
IVCXCASE	IONO3	INTEGER*4	4
RVCXH1P	VAR3	REAL*8	8
RVCXHBND	IONO3	REAL*8	8
RVCXHL	TEMP1	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RVCXHCT	OTHER	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXDR	PRAM	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RVCXL1	OTHER	REAL*8	8

Line# Source Line

```
      SUBROUTINE CASE3(RVSxH, IVSxJ)
C
C-------------------------------------------------------------
C
C   CASE3 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
C        IS IN THE CASE 3 PROFILE
C
C        CALLED BY: IONOPAR
C
C        CALLS:  PGVAL, PGF1L, PGF2, PGFB
C
C-------------------------------------------------------------
C
C        AUTHOR:          ERIC L. STROBEL & MICHAEL H. REILLY
C
C        DATE:            09/01/87
C
C        VERSION:         1.0
C
C-------------------------------------------------------------
C
C        REVISED:         09/01/87 -- V1.0.  Initial revision.
C
C-------------------------------------------------------------
C
C        USES:        RVSxH           The current height.
C
C            To determine which part of the CASE 3 profile is being
C         operated on.  The CASE 3 profile consists of (above the E.
C         the linear valley, a linear F1, the parabolic F2,
C         and the topside.
C
C        RETURNS:     IVSxJ           This helps distinguish which
C                                  boundary is being approached.
C
C-------------------------------------------------------------
C
      INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY
C
      REAL*8 RVSxH, RVCxH1L, RVCxHBND, RVCxHL, RVCxH2
      REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR
      REAL*8 RPCxRE, RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ
      REAL*8 RVCxRPI, RVCxV1, RVCxV2, RVCxXX, RVCxF40, RVCxF65
      REAL*8 RVCxHB1, RVCxYS, RVCxSL1, RVCxSL2, RVCxA2, RVCxK1
      REAL*8 RVCxB2, RVCxC2, RVCxHT3, RVCxHT4
C
      COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
      COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
      COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
      COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
```

Line#   Source Line

```
      COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
      COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RVCxSL1,RVCxSL2,RVCxH1L
      COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
C
      IF (RVSxH.LE.RVCxH1L) THEN
C
C     We're in the valley region, so go calculate the parameters
C       from the valley profile.
C
         CALL PGVAL(RVSxH)
         IF (IVCxSZ.GT.0) THEN
             RVCxHBND = RVCxH1L
             IVSxJ = 5
         ELSE
             RVCxHBND = RVCxHL
             IVSxJ = 4
         ENDIF
         RETURN
      ENDIF
      IF (RVSxH.LE.RVCxH2) THEN
C
C     We're in the F1 region, so go calculate the parameters
C       from the F1 profile.
C
         CALL PGF1L(RVSxH)
         IF (IVCxSZ.GT.0) THEN
             RVCxHBND = RVCxH2
             IVSxJ = 6
         ELSE
             RVCxHBND = RVCxH1L
             IVSxJ = 5
         ENDIF
         RETURN
      ENDIF
      IF (RVSxH.LE.RVCxHT5) THEN
C
C     We're in the F2 region, so go calculate the parameters
C       from the F2 profile.
C
         CALL PGF2(RVSxH)
         IF (IVCxSZ.GT.0) THEN
             RVCxHBND = RVCxHT5
             IVSxJ = 7
         ELSE
             RVCxHBND = RVCxH2
             IVSxJ =6
         ENDIF
         RETURN
      ENDIF
      IF (RVSxH.LE.RPCxHTP) THEN
```

Line# Source Line

```
C
C       We're in the topside region, so go calculate the parameters
C         from the topside profile.
C
              CALL PGFB(RVSxH)
              IF (IVCxSZ.GT.0) THEN
                  RVCxHBND = RPCxHTP
                  IVSxJ = 8
              ELSE
                  RVCxHBND = RVCxHT5
                  IVSxJ = 7
              ENDIF
          ELSE
C
C       We're beyond the cutoff of the ionosphere.
C
              IF (IVCxSZ.GT.0) THEN
                  RVCxHBND = RVCxHCT
              ELSE
                  RVCxHBND = RPCxHTP
              ENDIF
          ENDIF
C
C       Take care of things if the ray is headed toward a
C          boundary that's above the cutoff height.
C
          IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
              RVCxHBND = RVCxHCT
              IVSxJ = 1
          ENDIF
C
          RETURN
          END
```

CASE3 Local Symbols

Name	Class	Type	Size
IVSXJ	param		
RVSXH	param		
RVCXHCT	OTHER	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RPCXDR.	PRAM	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RVCXL1.	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXFPEQ.	OTHER	REAL*8	8
RVCXRPI	OTHER	REAL*8	8

CASE3 Local Symbols

Name	Class	Type	Size
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXXX.	IONO3	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXHB1	VAR2	REAL*8	8
RVCXYS.	VAR2	REAL*8	8
RVCXSL1	VAR2	REAL*8	8
RVCXSL2	VAR2	REAL*8	8
RVCXA2.	VAR4	REAL*8	8
RVCXK1.	TEMP1	REAL*8	8
RVCXB2.	VAR4	REAL*8	8
RVCXC2.	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
IVCXSZ.	MORE	INTEGER*4	4
IVCXCASE.	IONO3	INTEGER*4	4
IVCXSX.	MORE	INTEGER*4	4
IVCXSY.	MORE	INTEGER*4	4
RVCXH1L	VAR2	REAL*8	8
RVCXHBND.	IONO3	REAL*8	8
RVCXHL.	TEMP1	REAL*8	8
RVCXH2.	VAR2	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8

Line# Source Line

```
      SUBROUTINE CASE4(RVSxH, IVSxJ)
C
C-----------------------------------------------------------------
C
C   CASE4 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
C       IS IN THE CASE 4 PROFILE
C
C       CALLED BY: IONOPAR
C
C       CALLS:  PGVAL, PGF1P, PGF2, PGFB
C
C-----------------------------------------------------------------
C
C       AUTHOR:         ERIC L. STROBEL & MICHAEL H. REILLY
C
C       DATE:           09/01/87
C
C       VERSION:        1.0
C
C-----------------------------------------------------------------
C
C       REVISED:        09/01/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C       USES:       RVSxH           The current height.
C
C           To determine which part of the CASE 4 profile is being
C        operated on.  The CASE 4 profile consists of (above the E,
C        the linear valley, a portion of the F2, a parabolic F1,
C        the rest of the parabolic F2, and the topside.
C
C       RETURNS:    IVSxJ           This helps distinguish which
C                               boundary is being approached.
C
C-----------------------------------------------------------------
C
      INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY
C
      REAL*8 RVSxH, RVCxHU, RVCxHBND, RVCxHL, RVCxHT3, RVCxHT4
      REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR, RPCxRE
      REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RVCxRPI
      REAL*8 RVCxV1, RVCxV2, RVCxXX, RVCxA2, RVCxB2, RVCxC2
      REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU
      REAL*8 RTLx1, RTLx2
C
      COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
      COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
      COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
      COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
```

Line# Source Line

```
      COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
      COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RTLx1, RTLx2
      COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
C
      IF (RVSxH.LE.RVCxHU) THEN
C
C     We're in the valley region, so go calculate the parameters
C       from the valley profile.
C
          CALL PGVAL(RVSxH)
          IF (IVCxSZ.GT.0) THEN
              RVCxHBND = RVCxHU
              IVSxJ = 5
          ELSE
              RVCxHBND = RVCxHL
              IVSxJ = 4
          ENDIF
          RETURN
      ENDIF
      IF (RVSxH.LE.RVCxHT3) THEN
C
C     A portion of the F2 parabola is visible under the F1 layer.
C     (It's unknown how likely this is, but it at least appears
C     to be possible in the RADAR-C model, so it is safest to
C     include code for the possibility.)
C
          CALL PGF2(RVSxH)
          IF (IVCxSZ.GT.0) THEN
              RVCxHBND = RVCxHT3
              IVSxJ = 6
          ELSE
              RVCxHBND = RVCxHU
              IVSxJ = 5
          ENDIF
          RETURN
      ENDIF
      IF (RVSxH.LE.RVCxHT4) THEN
C
C     We're in the F1 region, so go calculate the parameters
C       from the F1 profile.
C
          CALL PGF1P(RVSxH)
          IF (IVCxSZ.GT.0) THEN
              RVCxHBND = RVCxHT4
              IVSxJ = 7
          ELSE
              RVCxHBND = RVCxHT3
              IVSxJ = 6
          ENDIF
          RETURN
```

Line# Source Line

```
            ENDIF
            IF (RVSxH.LE.RVCxHT5) THEN
C
C           We're in the more ordinary portion of the F2 region, so go
C           calculate the parameters from the F2 profile.
C
                CALL PGF2(RVSxH)
                IF (IVCxSZ.GT.0) THEN
                    RVCxHBND = RVCxHT5
                    IVSxJ = 8
                ELSE
                    RVCxHBND = RVCxHT4
                    IVSxJ = 7
                ENDIF
                RETURN
            ENDIF
            IF (RVSxH.LE.RPCxHTP) THEN
C
C           We're in the topside region, so go calculate the parameters
C             from the topside profile.
C
                CALL PGFB(RVSxH)
                IF (IVCxSZ.GT.0) THEN
                    RVCxHBND = RPCxHTP
                    IVSxJ = 9
                ELSE
                    RVCxHBND = RVCxHT5
                    IVSxJ = 8
                ENDIF
            ELSE
C
C           We're beyond the cutoff of the ionosphere.
C
                IF (IVCxSZ.GT.0) THEN
                    RVCxHBND = RVCxHCT
                ELSE
                    RVCxHBND = RPCxHTP
                ENDIF
            ENDIF
C
C           Take care of things if the ray is headed toward a
C              boundary that's above the cutoff height.
C
            IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
                RVCxHBND = RVCxHCT
                IVSxJ = 1
            ENDIF
C
            RETURN
            END
```

CASE4 Local Symbols

Name	Class	Type	Size
IVSXJ	param		
RVSXH	param		
RVCXHU.	VAR1	REAL*8	8
RVCXHBND.	IONO3	REAL*8	8
RVCXHL.	TEMP1	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RVCXHCT	OTHER	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RPCXDR.	PRAM	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RVCXL1.	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXFREQ.	OTHER	REAL*8	8
RVCXRPI	OTHER	REAL*8	8
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXXX.	IONO3	REAL*8	8
RVCXA2.	VAR4	REAL*8	8
RVCXB2.	VAR4	REAL*8	8
RVCXC2.	VAR4	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXK1.	TEMP1	REAL*8	8
RVCXXL.	VAR1	REAL*8	8
RVCXXU.	VAR1	REAL*8	8
RTLX1	VAR1	REAL*8	8
RTLX2	VAR1	REAL*8	8
IVCXSZ.	MORE	INTEGER*4	4
IVCXCASE.	IONO3	INTEGER*4	4
IVCXSX.	MORE	INTEGER*4	4
IVCXSY.	MORE	INTEGER*4	4

Line# Source Line

```
      SUBROUTINE CASE5(RVSxH, IVSxJ)
C
C-----------------------------------------------------------------------
C
C   CASE5 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
C         IS IN THE CASE 5 PROFILE
C
C         CALLED BY: IONOPAR
C
C         CALLS:  PGVAL, PGF1P, PGF2, PGFB
C
C-----------------------------------------------------------------------
C
C         AUTHOR:           ERIC L. STROBEL & MICHAEL H. REILLY
C
C         DATE:             09/01/87
C
C         VERSION:          1.0
C
C-----------------------------------------------------------------------
C
C         REVISED:          09/01/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------------
C
C         USES:          RVSxH          The current height.
C
C            To determine which part of the CASE 5 profile is being
C          operated on.  The CASE 5 profile consists of (above the E)
C          the linear valley, a parabolic F1, an additional valley
C          segment, the parabolic F2, and the topside.
C
C         RETURNS:       IVSxJ          This helps distinguish which
C                                   boundary is being approached.
C
C-----------------------------------------------------------------------
C
      INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY
C
      REAL*8 RVSxH, RVCxHU, RVCxHBND, RVCxHL, RVCxHT3, RVCxHT4
      REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR, RPCxRE
      REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RVCxRPI
      REAL*8 RVCxV1, RVCxV2, RVCxXX, RVCxA2, RVCxB2, RVCxC2
      REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU
      REAL*8 RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
      REAL*8 RTLx1, RTLx2
C
      COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
      COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
      COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
```

Line# Source Line

```
      COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
      COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
      COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RTLx1, RTLx2
      COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
      COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
C
      IF (RVSxH.LE.RVCxH1P) THEN
C
C     We're in the valley region, so go calculate the parameters
C       from the valley profile.
C
         CALL PGVAL(RVSxH)
         IF (IVCxSZ.GT.0) THEN
            RVCxHBND = RVCxH1P
            IVSxJ = 5
         ELSE
            RVCxHBND = RVCxHL
            IVSxJ = 4
         ENDIF
         RETURN
      ENDIF
      IF (RVSxH.LE.RVCxH2P) THEN
C
C     We're in the F1 region, so go calculate the parameters
C       from the F1 profile.
C
         CALL PGF1P(RVSxH)
         IF (IVCxSZ.GT.0) THEN
            RVCxHBND = RVCxH2P
            IVSxJ = 6
         ELSE
            RVCxHBND = RVCxH1P
            IVSxJ = 5
         ENDIF
         RETURN
      ENDIF
      IF (RVSxH.LE.RVCxHU) THEN
C
C     Surprise! A portion of the valley profile is seen above
C       the F1. And yes, this really can in fact happen in the
C       RADAR-C model.
C
         CALL PGVAL(RVSxH)
         IF (IVCxSZ.GT.0) THEN
            RVCxHBND = RVCxHU
            IVSxJ = 7
         ELSE
            RVCxHBND = RVCxH2P
            IVSxJ = 6
         ENDIF
```

```
Line#   Source Line

                RETURN
            ENDIF
            IF (RVSxH.LE.RVCxHT5) THEN
C
C           We're in the F2 region, so go calculate the parameters
C             from the F2 profile.
C
                CALL PGF2(RVSxH)
                IF (IVCxSZ.GT.0) THEN
                    RVCxHBND = RVCxHT5
                    IVSxJ = 8
                ELSE
                    RVCxHBND = RVCxHU
                    IVSxJ = 7
                ENDIF
                RETURN
            ENDIF
            IF (RVSxH.LE.RPCxHTP) THEN
C
C           We're in the topside region, so go calculate the parameters
C             from the topside profile.
C
                CALL PGFB(RVSxH)
                IF (IVCxSZ.GT.0) THEN
                    RVCxHBND = RPCxHTP
                    IVSxJ = 9
                ELSE
                    RVCxHBND = RVCxHT5
                    IVSxJ = 8
                ENDIF
            ELSE
C
C           We're beyond the cutoff of the ionosphere.
C
                IF (IVCxSZ.GT.0) THEN
                    RVCxHBND = RVCxHCT
                ELSE
                    RVCxHBND = RPCxHTP
                ENDIF
            ENDIF
C
C           Take care of things if the ray is headed toward a
C              boundary that's above the cutoff height.
C
            IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
                RVCxHBND = RVCxHCT
                IVSxJ = 1
            ENDIF
C
            RETURN
```

```
Line#   Source Line

 1320           END
```

CASE5 Local Symbols

Name	Class	Type	Size
IVSXJ	param		
RVSXH	param		
IVCXSZ.	MORE	INTEGER*4	4
IVCXCASE.	IONO3	INTEGER*4	4
IVCXSX.	MORE	INTEGER*4	4
IVCXSY.	MORE	INTEGER*4	4
RVCXHU.	VAR1	REAL*8	8
RVCXHBND.	IONO3	REAL*8	8
RVCXHL.	TEMP1	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RVCXHCT	OTHER	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RPCXDR.	PRAM	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RVCXL1.	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXFREQ.	OTHER	REAL*8	8
RVCXRPI	OTHER	REAL*8	8
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXXX.	IONO3	REAL*8	8
RVCXA2.	VAR4	REAL*8	8
RVCXB2.	VAR4	REAL*8	8
RVCXC2.	VAR4	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXK1.	TEMP1	REAL*8	8
RVCXXL.	VAR1	REAL*8	8
RVCXXU.	VAR1	REAL*8	8
RVCXA1.	VAR3	REAL*8	8
RVCXB1.	VAR3	REAL*8	8
RVCXC1.	VAR3	REAL*8	8
RVCXH1P	VAR3	REAL*8	8
RVCXH2P	VAR3	REAL*8	8
RTLX1	VAR1	REAL*8	8
RTLX2	VAR1	REAL*8	8

Line# Source Line

```
      SUBROUTINE PGVAL(RVSxH)
C
C-----------------------------------------------------------------
C
C    PGVAL -- SUBROUTINE to calculate the ionospheric parameters
C         for the linear valley profile.
C
C
C       CALLED BY: CASE1, CASE2, CASE3, CASE4, CASE5
C
C-----------------------------------------------------------------
C
C       AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
C
C       DATE:            09/01/87
C
C       VERSION:         1.0
C
C-----------------------------------------------------------------
C
C       REVISED:         09/01/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C       USES:            RVSxH        The current height.
C
C
C       To calculate the A, N1, and N2 parameters, as well as the
C         plasma frequency squared XX.  For more details, see the
C         Radio Science paper and the RADAR-C report referred to
C         in the documentation.
C
C
C       RETURNS:         RVCxALPH     A coef. in the index of
C                                      refraction.
C
C                        RVCxET1      A coef. in the index of
C                                      refraction.
C
C                        RVCxET2      A coef. in the index of
C                                      refraction.
C
C                        RVCxXX       The plasma frequency squared.
C
C
C-----------------------------------------------------------------
C
      INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
      INTEGER ITCxA
C
```

```
Line#   Source Line

              REAL*8 RVSxH, RVCxXX, RVCxA0, RVCxB0, RVCxALPH, RVCxFREQ
              REAL*8 RVLxHUXE, RVCxHB(3), RVCxFB(3), RVCxHU, RVLxHTR
              REAL*8 RVCxHT3, RVCxHL, RVLxXXXE, RVLxHUX2, RVLxXXH2
              REAL*8 RVLxXXX2, RVLxHUY2, RVLxXXY2, RVCxET1, RVCxB5(3)
              REAL*8 RVCxB6(3), RVCxV1, RVCxET2, RVCxE5(3), RVCxE6(3)
              REAL*8 RVCxV2, RVCxL1, RVCxLO1, RVCxHBOT
              REAL*8 RVCxBETA, RVCxA5(3), RVCxA6(3), RVCxF40, RVCxC2
              REAL*8 RVCxF65, RVCxK1, RVCxXL, RVCxXU, RVCxA2, RVCxB2
              REAL*8 RTCxA, RTCxB, RTCxC, RTCxD, RTCxE
      C
              COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
              COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
              COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
              COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
              COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
              COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
              COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
              COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RTCxD,RTCxE
      C
              RVCxXX = RVCxA0 + RVCxB0 * RVSxH
              RVCxALPH = RVCxB0 / RVCxFREQ
              IF (IFCxN4.NE.1) THEN
                  RVLxHUXE = (0.98D00*RVCxHB(3)) * (0.98D00*RVCxHB(3))
                  RVLxHUXE = RVLxHUXE / (2.0D00*RVCxFB(3)*(RVCxHB(2)-RVCxHU)
                  RVLxHTR = (RVCxHT3 - RVCxHL)/(RVCxHU - RVCxHL)
                  RVLxXXXE = RVCxXX/RVCxFB(1) - RVCxB0*RVLxHTR*RVLxHUXE
                  RVLxHUX2 = -RVLxHUXE * RVCxFB(1 /RVCxFB(3)
                  RVLxXXH2 = -RVCxB0 * RVLxHTR
                  RVLxXXX2 = RVLxXXH2 * RVLxHUX2
                  RVLxHUY2 = -(RVCxHB(2) - RVCxHU) / RVCxHB(3)
                  RVLxXXY2 = RVLxXXH2 * RVLxHUY2
                  RVCxET1 = RVLxXXXE*RVCxB5(1) + RVLxXXX2*RVCxB5(3)
                  RVCxET1 = RVCxET1 + RVLxXXH2*RVCxB6(2) + RVLxXXY2*RVCxB6 (
                  RVCxET1 = -RVCxET1 / (RVCxV1 * RVCxFREQ)
                  RVCxET2 = RVLxXXXE*RVCxE5(1) + RVLxXXX2*RVCxE5(3)
                  RVCxET2 = RVCxET2 + RVLxXXH2*RVCxE6(2) + RVLxXXY2*RVCxE6 (
                  RVCxET2 = RVCxET2 / (RVCxV2 * RVCxFREQ)
              ENDIF
              RETURN
              END
```

PGVAL Local Symbols

Name	Class	Type	Size
RVSXH	param		
RVLXXXX2.	local	REAL*8	8
RVLXHUXE.	local	REAL*8	8
RVLXXXY2.	local	REAL*8	8

PGVAL Local Symbols

Name	Class	Type	Size
RVLXXXXE.	local	REAL*8	8
RVLXHUX2.	local	REAL*8	8
RVLXHUY2.	local	REAL*8	8
RVLXXXH2.	local	REAL*8	8
RVLXHTR	local	REAL*8	8
RVCXA2.	VAR4	REAL*8	8
RVCXB2.	VAR4	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	IONO3	REAL*8	8
RTCXD	VAR4	REAL*8	8
RTCXE	VAR4	REAL*8	8
IFCXN4.	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1.	TEMP2	INTEGER*4	4
IVCXSCT2.	TEMP2	INTEGER*4	4
IVCXSCT3.	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXXX.	IONO3	REAL*8	8
RVCXA0.	VAR1	REAL*8	8
RVCXB0.	VAR1	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXFREQ.	OTHER	REAL*8	8
RVCXHB.	IONO1	REAL*8	24
RVCXFB.	IONO1	REAL*8	24
RVCXHU.	VAR1	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHL.	TEMP1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV2.	IONO3	REAL*8	8
RVCXL1.	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXF40	TEMP1	REAL*8	8
RVCXC2.	VAR4	REAL*8	8
RVCXF65	TEMP1	REAL*8	8

PGVAL Local Symbols

Name	Class	Type	Size
RVCXK1.	TEMP1	REAL*8	8
RVCXXL.	VAR1	REAL*8	8
RVCXXU.	VAR1	REAL*8	8

Line# Source Line

```
      SUBROUTINE PGF1L(RVSxH)
C
C-----------------------------------------------------------------
C
C    PGF1L -- SUBROUTINE to calculate the ionospheric parameters
C             for the linear F1 region.
C
C
C       CALLED BY: CASE3
C
C-----------------------------------------------------------------
C
C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C       DATE:           09/01/87
C
C       VERSION:        1.0
C
C-----------------------------------------------------------------
C
C       REVISED:        09/01/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C       USES:           RVSxH        The current height.
C
C
C       To calculate the A, N1, and N2 parameters, as well as the
C         plasma frequency squared XX.  For more details, see the
C         Radio Science paper and the RADAR-C report referred to
C         in the documentation.
C
C
C       RETURNS:        RVCxALPH     A coef. in the index of
C                                     refraction.
C
C                       RVCxET1      A coef. in the index of
C                                     refraction.
C
C                       RVCxET2      A coef. in the index of
C                                     refraction.
C
C                       RVCxXX       The plasma frequency squared.
C
C
C-----------------------------------------------------------------
C
      INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
      INTEGER ITCxA
C
```

```
Line#  Source Line

        REAL*8 RVCxXX, RVCxSL1, RVLxHTV, RVCxALPH, RVCxFREQ
        REAL*8 RVCxHB1, RVCxYS, RVLxS1X1, RVLxS1H1, RVLxYSX1
        REAL*8 RVCxHB(3), RVCxFB(3), RVCxH2, RVLxXXX1, RVLxXXH1
        REAL*8 RVCxET1, RVCxET2, RVCxB5(3), RVCxB6(3), RVCxE5(3)
        REAL*8 RVCxE6(3), RVCxV1, RVCxV2, RVLxS1X2, RVLxXXX2
        REAL*8 RVLxXXH2, RVLxYSY2, RVLxS1Y2, RVLxXXY2, RVCxA5(3)
        REAL*8 RVCxA6(3),RVCxL1,RVCxLO1,RVCxHBOT,RVCxBETA,RVSxH
        REAL*8 RTCxA, RTCxB, RTCxC, RTCxD, RTCxE
C
        COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
        COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
        COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
        COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
        COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
        COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RVCxSL1,RTCxD,RTCxE
C
        RVLxHTV = RVSxH - RVCxHB1
        RVCxXX = RVCxSL1 * RVLxHTV
        RVCxALPH = RVCxSL1 / RVCxFREQ
        IF (IFCxN4.EQ.1) RETURN
        IF (RVCxYS.EQ.1) THEN
            RVLxS1X1 = 1.0D00
            RVLxS1H1 = 0.0D00
        ELSE
            RVLxYSX1 = RVCxHB(3)*RVCxHB(3)/(2.0D00*RVCxFB(3)*
     *(RVCxHB(2) - RVCxH2))
            RVLxS1X1 = (1.0D00 - (RVCxFB(2)*RVLxYSX1/RVCxYS)) / RVCxYS
            RVLxS1H1 = RVCxFB(2) / (RVCxYS*RVCxYS)
        ENDIF
        RVLxXXX1 = RVLxHTV * RVLxS1X1
        RVLxXXH1 = 0.75D00*(-RVCxSL1 + RVLxHTV*RVLxS1H1)
        RVCxET1 = -RVLxXXX1*RVCxB5(2) - RVLxXXH1*RVCxB6(2)
        RVCxET2 = RVLxXXX1 * RVCxE5(2) + RVLxXXH1*RVCxE6(2)
        IF (RVCxYS.EQ.1) THEN
            RVCxET1 = RVCxET1 / (RVCxV1 * RVCxFREQ)
            RVCxET2 = RVCxET2 / (RVCxV2 * RVCxFREQ)
            RETURN
        ENDIF
        RVLxS1X2 = RVLxS1H1 * RVLxYSX1 * RVCxFB(2)/ RVCxFB(3)
        RVLxXXX2 = RVLxHTV * RVLxS1X2
        RVLxXXH2 = -RVLxHTV * RVLxS1H1
        RVLxYSY2 = (RVCxH2 - RVCxHB(2)) / RVCxHB(3)
        RVLxS1Y2 = -RVLxS1H1 * RVLxYSY2
        RVLxXXY2 = RVLxHTV * RVLxS1Y2
        RVCxET1 = RVCxET1 - RVLxXXX2*RVCxB5(3) - RVLxXXH2*RVCxB6(3)
        RVCxET1 = (RVCxET1 - RVLxXXY2*RVCxB6(3)) / (RVCxV1*RVCxFREQ)
        RVCxET2 = RVCxET2 + RVLxXXX2*RVCxE5(3) + RVLxXXH2*RVCxE6(3)
        RVCxET2 = (RVCxET2 + RVLxXXY2*RVCxE6(3)) / (RVCxV2*RVCxFREQ)
        RETURN
        END
```

PGF1L Local Symbols

Name	Class	Type	Size
RVSXH	param		
RVLXYSX1.	local	REAL*8	8
RVLXYSY2.	local	REAL*8	8
RVLXXXX1.	local	REAL*8	8
RVLXXXX2.	local	REAL*8	8
RVLXXXY2.	local	REAL*8	8
RVLXS1H1.	local	REAL*8	8
RVLXS1X1.	local	REAL*8	8
RVLXS1X2.	local	REAL*8	8
RVLXS1Y2.	local	REAL*8	8
RVLXXXH1.	local	REAL*8	8
RVLXXXH2.	local	REAL*8	8
RVLXHTV	local	REAL*8	8
IFCXN4.	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1.	TEMP2	INTEGER*4	4
IVCXSCT2.	TEMP2	INTEGER*4	4
IVCXSCT3.	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXXX.	IONO3	REAL*8	8
RVCXSL1	VAR2	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXFREQ.	OTHER	REAL*8	8
RVCXHB1	VAR2	REAL*8	8
RVCXYS.	VAR2	REAL*8	8
RVCXHB.	IONO1	REAL*8	24
RVCXFB.	IONO1	REAL*8	24
RVCXH2.	VAR2	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXL1.	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8

PGF1L Local Symbols

Name	Class	Type	Size
RTCXC	IONO3	REAL*8	8
RTCXD	VAR2	REAL*8	8
RTCXE	VAR2	REAL*8	8

Line# Source Line

```
      SUBROUTINE PGF1P(RVSxH)
C
C-----------------------------------------------------------------
C
C    PGF1P -- SUBROUTINE to calculate the ionospheric parameters
C           for the parabolic F1 region.
C
C
C       CALLED BY: CASE2, CASE4, CASE5
C
C-----------------------------------------------------------------
C
C       AUTHOR:           MICHAEL H. REILLY & ERIC L. STROBEL
C
C       DATE:             09/01/87
C
C       VERSION:          1.0
C
C-----------------------------------------------------------------
C
C       REVISED:          09/01/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C       USES:             RVSxH        The current height.
C
C
C       To calculate the A, B, N1, and N2 parameters, as well as the
C         plasma frequency squared XX.  For more details, see the
C         Radio Science paper and the RADAR-C report referred to
C         in the documentation.
C
C
C       RETURNS:          RVCxALPH     A coef. in the index of
C                                       refraction.
C
C                         RVCxBETA     A coef. in the index of
C                                       refraction.
C
C                         RVCxET1      A coef. in the index of
C                                       refraction.
C
C                         RVCxET2      A coef. in the index of
C                                       refraction.
C
C                         RVCxXX       The plasma frequency squared.
C
C
C-----------------------------------------------------------------
C
```

Line# Source Line

```
          INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
          INTEGER ITCxA
C
          REAL*8 RVSxH, RVLxYF1, RVLxHTV, RVCxHB(3), RVCxXX
          REAL*8 RVCxFB(3), RVCxFREQ, RVCxALPH, RVCxBETA, RVLxXXX1
          REAL*8 RVLxXXH1, RVCxET1, RVCxET2, RVCxB5(3), RVCxB6(3)
          REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxL1, RVCxLO1
          REAL*8 RVCxHBOT, RVCxA5(3), RVCxA6(3), RTCxA, RTCxB
          REAL*8 RTCxC
C
          COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
          COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
          COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
          COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
          COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
C
          RVLxYF1 = 0.25D00 * RVCxHB(1)
          RVLxHTV = (RVCxHB(1) - RVSxH) / RVLxYF1
          RVCxXX = RVCxFB(2) * (1.0D00 - RVLxHTV*RVLxHTV)
          RVCxALPH = 2.0D00 * RVCxFB(2) * RVLxHTV / (RVLxYF1 * RVCxFREQ)
          RVCxBETA = RVCxALPH / (2.0D00 * RVLxHTV * RVLxYF1)
          IF (IFCxN4.EQ.1) RETURN
          RVLxXXX1 = RVCxXX / (RVCxFB(2) * RVCxFREQ)
          RVLxXXH1 = -2.0D00 * RVCxBETA * RVSxH *
     #(1.0D00 - RVSxH/RVCxHB(1))
          RVCxET1 = RVLxXXX1*RVCxB5(2) + RVLxXXH1*RVCxB6(1)
          RVCxET1 = -RVCxET1 / RVCxV1
          RVCxET2 = RVLxXXX1*RVCxE5(2) + RVLxXXH1*RVCxE6(1)
          RVCxET2 = RVCxET2 / RVCxV2
          RETURN
          END
```

PGF1P Local Symbols

Name	Class	Type	Size
RVSXH	param		
RVLXXXX1.	local	REAL*8	8
RVLXYF1	local	REAL*8	8
RVLXXXH1.	local	REAL*8	8
RVLXHTV	local	REAL*8	8
IFCXN4.	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1.	TEMP2	INTEGER*4	4
IVCXSCT2.	TEMP2	INTEGER*4	4
IVCXSCT3.	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXHB.	IONO1	REAL*8	24
RVCXXX.	IONO3	REAL*8	8

PGF1P Local Symbols

Name	Class	Type	Size
RVCXFB.	IONO1	REAL*8	24
RVCXFREQ.	OTHER	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXL1.	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	IONO3	REAL*8	8

Line# Source Line

```
      SUBROUTINE PGF2(RVSxH)
C
C-------------------------------------------------------------------
C
C     PGF2 -- SUBROUTINE to calculate the ionospheric parameters
C             for the parabolic F2 region.
C
C
C       CALLED BY: CASE1, CASE2, CASE3, CASE4, CASE5
C
C-------------------------------------------------------------------
C
C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C       DATE:           09/01/87
C
C       VERSION:        1.0
C
C-------------------------------------------------------------------
C
C       REVISED:        09/01/87 -- V1.0.  Initial revision.
C
C-------------------------------------------------------------------
C
C       USES:           RVSxH         The current height.
C
C
C       To calculate the A, B, N1, and N2 parameters, as well as the
C         plasma frequency squared XX.  For more details, see the
C         Radio Science paper and the RADAR-C report referred to
C         in the documentation.
C
C
C       RETURNS:        RVCxALPH      A coef. in the index of
C                                      refraction.
C
C                       RVCxBETA      A coef. in the index of
C                                      refraction.
C
C                       RVCxET1       A coef. in the index of
C                                      refraction.
C
C                       RVCxET2       A coef. in the index of
C                                      refraction.
C
C                       RVCxXX        The plasma frequency squared.
C
C
C-------------------------------------------------------------------
C
```

```
Line#  Source Line

             INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
             INTEGER ITCxA
       C
             REAL*8 RVSxH, RVLxHTV, RVCxHB(3), RTLx1, RVCxXX
             REAL*8 RVCxFB(3), RVCxFREQ, RVCxALPH, RVCxBETA
             REAL*8 RVLxXXX2, RVLxXXH2, RVLxXXY2, RVCxET1, RVCxET2
             REAL*8 RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxE6(3)
             REAL*8 RVCxV1, RVCxV2, RVCxL1, RVCxLO1, RVCxHBOT
             REAL*8 RVCxA5(3), RVCxA6(3), RTCxA, RTCxB, RTCxC
       C
             COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
             COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxH
             COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
             COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
             COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
       C
             RVLxHTV = (RVCxHB(2) - RVSxH) / RVCxHB(3)
             RTLx1 = 1.0D00 - RVLxHTV*RVLxHTV
             RVCxXX = RVCxFB(3) * RTLx1
             RVCxBETA = RVCxFB(3) / (RVCxHB(3) * RVCxHB(3) * RVCxFREQ)
             RVCxALPH = 2.0D00 * RVCxBETA * ( RVCxHB(2) - RVSxH )
             IF (IFCxN4.EQ.1) RETURN
             RVLxXXX2 = RTLx1
             RVLxXXH2 = -2.0D00 * RVCxFB(3) * RVLxHTV / RVCxHB(3)
             RVLxXXY2 = -RVLxXXH2 * RVLxHTV
             RVCxET1 = RVLxXXX2*RVCxB5(3) + RVLxXXH2*RVCxB6(2)
             RVCxET1 = -(RVCxET1 + RVLxXXY2*RVCxB6(3))/(RVCxV1*RVCxFREQ)
             RVCxET2 = RVLxXXX2*RVCxE5(3) + RVLxXXH2*RVCxE6(2)
             RVCxET2 = (RVCxET2 + RVLxXXY2*RVCxE6(3))/(RVCxV2*RVCxFREQ)
             RETURN
             END
```

PGF2 Local Symbols

Name	Class	Type	Size
RVSXH	param		
RVLXXXX2.	local	REAL*8	8
RVLXXXY2.	local	REAL*8	8
RVLXXXH2.	local	REAL*8	8
RTLX1	local	REAL*8	8
RVLXHTV	local	REAL*8	8
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXL1.	OTHER	REAL*8	8

PGF2 Local Symbols

Name	Class	Type	Size
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	IONO3	REAL*8	8
IFCXN4.	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1.	TEMP2	INTEGER*4	4
IVCXSCT2.	TEMP2	INTEGER*4	4
IVCXSCT3.	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXHB.	IONO1	REAL*8	24
RVCXXX.	IONO3	REAL*8	8
RVCXFB.	IONO1	REAL*8	24
RVCXFREQ.	OTHER	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8

Line# Source Line

```
      SUBROUTINE PGFB(RVSxH)
C
C-----------------------------------------------------------------
C
C    PGFB -- SUBROUTINE to calculate the ionospheric parameters
C            for the Bent topside region.
C
C
C       CALLED BY: CASE1, CASE2, CASE3, CASE4, CASE5
C
C-----------------------------------------------------------------
C
C       AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
C
C       DATE:            09/01/87
C
C       VERSION:         1.0
C
C-----------------------------------------------------------------
C
C       REVISED:         09/01/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C       USES:            RVSxH         The current height.
C
C
C       To calculate the A, B, N1, and N2 parameters, as well as the
C         plasma frequency squared XX.  For more details, see the
C         Radio Science paper and the RADAR-C report referred to
C         in the documentation.
C
C
C       RETURNS:         RVCxALPH      A coef. in the index of
C                                       refraction.
C
C                        RVCxBETA      A coef. in the index of
C                                       refraction.
C
C                        RVCxET1       A coef. in the index of
C                                       refraction.
C
C                        RVCxET2       A coef. in the index of
C                                       refraction.
C
C                        RVCxXX        The plasma frequency squared.
C
C
C-----------------------------------------------------------------
C
```

Line# Source Line

```
      INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
      INTEGER ITCxA
C
      REAL*8 RVLxALP, RVCxHB(3), RVLxA7, RVCxFB(3), RVSxH
      REAL*8 RVCxHT5, RVCxXX, RVCxALPH, RVCxBETA, RVCxFREQ
      REAL*8 RVLxXXX2, RVLxXXH2, RVLxXXY2, RVCxET1, RVCxET2
      REAL*8 RVCxA5(3), RVCxA6(3), RVCxB5(3), RVCxB6(3)
      REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxL1, RVCxLO1
      REAL*8 RVCxHBOT, RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4
      REAL*8 RTCxA, RTCxB, RTCxC
C
      COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
      COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
      COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
      COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
      COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
      COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
C
      RVLxALP = 1.0D00 / (1.875D00 * RVCxHB(3))
      RVLxA7 = 15.0D00 * RVCxFB(3) * DEXP(RVLxALP*RVCxHT5)/16.0D00
      RVCxXX = RVLxA7 * DEXP(-RVLxALP * RVSxH)
      RVCxALPH = -RVLxALP * RVCxXX / RVCxFREQ
      RVCxBETA = RVLxALP * RVCxALPH / 2.0D00
      IF (IFCxN4.EQ.1) RETURN
      RVLxXXX2 = RVCxXX / (RVCxFB(3) * RVCxFREQ)
      RVLxXXH2 = -RVCxALPH
      RVLxXXY2 = -RVCxALPH * (0.25D00 - 1.375D00*RVLxALP*
     #(RVCxHT5-RVSxH))
      RVCxET1 = RVLxXXX2*RVCxB5(3) + RVLxXXH2*RVCxB6(2)
      RVCxET1 = -(RVCxET1 + RVLxXXY2*RVCxB6(3)) / RVCxV1
      RVCxET2 = RVLxXXX2*RVCxE5(3) + RVLxXXH2*RVCxE6(2)
      RVCxET2 = (RVCxET2 + RVLxXXY2*RVCxE6(3)) / RVCxV2
      RETURN
      END
```

PGFB Local Symbols

Name	Class	Type	Size
RVSXH	param		
RVLXA7.	local	REAL*8	8
RVLXXXX2.	local	REAL*8	8
RVLXXXY2.	local	REAL*8	8
RVLXALP	local	REAL*8	8
RVLXXXH2.	local	REAL*8	8
IFCXN4.	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1.	TEMP2	INTEGER*4	4
IVCXSCT2.	TEMP2	INTEGER*4	4

PGFB Local Symbols

Name	Class	Type	Size
IVCXSCT3.	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXHB.	IONO1	REAL*8	24
RVCXFB.	IONO1	REAL*8	24
RVCXHT5	VAR4	REAL*8	8
RVCXXX.	IONO3	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXFREQ.	OTHER	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXL1.	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT.	OTHER	REAL*8	8
RVCXA2.	VAR4	REAL*8	8
RVCXB2.	VAR4	REAL*8	8
RVCXC2.	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	IONO3	REAL*8	8

Global Symbols

Name	Class	Type	Size
CASE1	FSUBRT	***	***
CASE2	FSUBRT	***	***
CASE3	FSUBRT	***	***
CASE4	FSUBRT	***	***
CASE5	FSUBRT	***	***
END	common	***	32
INBOX	extern	***	***
INTERP.	FSUBRT	***	***
INTSIGN	extern	INTEGER*4	***

Global Symbols

Name	Class	Type	Size
IONO1	common	***	30
IONO2	common	***	144
IONO3	common	***	36
IONOPAR	FSUBRT	***	***
MAINDAT	common	***	48
MORE.	common	***	12
OTHER	common	***	48
PGF1L	FSUBRT	***	***
PGF1P	FSUBRT	***	***
PGF2.	FSUBRT	***	***
PGFB.	FSUBRT	***	***
PGVAL	FSUBRT	***	***
PRAM.	common	***	32
RAID.	common	***	4
SCPS1	common	***	28804
SCPS1A.	common	***	86400
START	common	***	40
TEMP1	common	***	32
TEMP2	common	***	20
TEMP3	common	***	32
TRIANG.	extern	***	***
VAR1.	common	***	40
VAR2.	common	***	48
VAR3.	common	***	40
VAR4.	common	***	48

Code size = 356f (13679)
Data size = 019f (415)

Line# Source Line

```
      SUBROUTINE ENDPT(IFSxG, RVSxH0, RVSxHZ, RVSxS4, RVSxGPI)
C
C-------------------------------------------------------------
C
C  ENDPT -- SUBROUTINE TO CALCULATE THE ENDPOINTS OF THE INTERVAL
C
C        CALLED BY:   RAYSUB
C
C        CALLS:       ACCFSP
C
C-------------------------------------------------------------
C
C        AUTHOR:         MICHAEL H. REILLY
C
C        DATE:           07/25/86
C
C        VERSION:        1.1
C
C-------------------------------------------------------------
C
C        REVISED:        07/25/86 -- INITIAL REVISION.  TRANSLATED
C                        FROM TEKTRONIX BASIC TO VAX FORTRAN BY
C                        ERIC L. STROBEL.
C
C                        07/30/86 -- V1.1.  Change over to use of
C                        REAL*8 precision in the calculations.
C
C                        09/01/87 -- V2.0.  Changed to prevent
C                        problems with calculations on small numbers.
C
C-------------------------------------------------------------
C
C       USES:  IFSxG           A FLAG
C              RVSxH0          INITIAL HEIGHT
C              RVSxHZ          HEIGHT
C              RVCxH5          NEW HEIGHT
C              RVCxALPH        X
C              RVCxBETA        X
C              RVCxET1         X-IONOSPHERIC PARAMETERS
C              RVCxET2         X
C              RVCxHB          HEIGHT OF IONOSPHERE BOTTOM
C              RVCxCX,CY,CZ
C              IVCxSX,SY,SZ
C
C        TO COMPUTE THE S.E.Z. COORDINATES FOR THE END OF THE
C              RAYPATH INCREMENT. (See the Radio Science paper
C              referred to in the documentation.)
C
C        RETURNS:
C              IFSxG
```

```
Line#  Source Line

C                 RVSxGPI          GROUP PATH INCREMENT
C                 RVCxXF.YF.ZF     COORDINATES OF THE ENDPOINT
C
C-------------------------------------------------------------
C
        LOGICAL LTLxT1, LTLxT2, LTLxT3, LTLxT4
C
        INTEGER IFSxG, IVCxSX, IVCxSY, IVCxSZ
C
        REAL*8 RVCxCX, RVCxCY, RVCxCZ, RVCxLAT0, RVCxLON0
        REAL*8 RVCxHB, RVCxXF, RVCxYF, RVCxZF, RTLxTEMP
        REAL*8 RVCxH5, RVSxH0, RVSxHZ, RVSxGPI, RVCxALPH
        REAL*8 RVCxBETA, RVCxET1, RVCxET2, RVLxEPS
        REAL*8 RTLxF3, RTLxR2, RTLxF6, RTLxF4, RTLxB1
        REAL*8 RTLxA2, RTLxF5, RTLxG3, RTLxC4, RTLxC5, RTLxF1
        REAL*8 RTLxG1, RTLxC6, RTLxC7, RTLxF2, RTLxG2
        REAL*8 RPCxPI, RPCxDTR, RPCxRE, RPCxHTP, RVSxS4
C
        COMMON /OTHER/ RVCxLAT0,RVCxLON0,RVCxHB
        COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RVCxH5
        COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2
        COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ, RVCxCX, RVCxCY, RVCxCZ
        COMMON /PRAM/ RPCxPI, RPCxDTR, RPCxRE, RPCxHTP
        COMMON /TEMP4/ RTLxF1, RTLxF2, RTLxF3
C
C       Initializations.
C
        RTLxA2 = 0.0
        RTLxB1 = 0.0
        RTLxC4 = 0.0
        RTLxC5 = 0.0
        RTLxC6 = 0.0
        RTLxC7 = 0.0
        RTLxF4 = 0.0
        RTLxF5 = 0.0
        RTLxF6 = 0.0
        RTLxG1 = 0.0
        RTLxG2 = 0.0
        RTLxG3 = 0.0
        RTLxR2 = 0.0
        RVCxXF = 0.0D00
        RVCxYF = 0.0D00
        RVSxGPI = 0.0D00
        RVLxEPS = 1.0D-09
C
C       Check for being in free space.
C
        IF (RVCxH5.GE.(RVCxHB - 0.02D00).AND.RVCxH5.LE.
     #(RPCxHTP + 0.02D00)) GO TO 10262
        IF (IVCxSZ.GE.0) THEN
```

Line#   Source Line

```
                IFSxG = 7
            ELSE
                IFSxG = 6
            ENDIF
      C
      C     ACCFSP is the routine that does the free space propagation.
      C
            CALL ACCFSP(RVSxH0,RVSxS4,IFSxG)
            GO TO 10267
      10262 RVCxZF = RVSxHZ - RVSxH0
      C
      C     Try to prevent loss of precision in the quantity
      C        (sqrt(x) - sqrt(x+y)) with y small.  First, define
      C        'small', then round the offending terms to zero.
      C
      10263 IF (RVCxBETA.NE.0.0D00) THEN
                LTLxT1 = DABS(RVCxBETA*RVCxZF*RVCxZF).LT.(RVLxEPS*
           #RVCxCZ)
                LTLxT2 = DABS(RVCxALPH*RVCxZF).LT.(RVLxEPS*RVCxCZ)
                IF (LTLxT1.AND.LTLxT2) THEN
                    RVCxALPH = 0.0D00
                    RVCxBETA = 0.0D00
                    GO TO 10267
                ENDIF
                IF (IFSxG.EQ.3) GO TO 10282
                GO TO 10280
            ENDIF
      10265 IF (RVCxALPH.NE.0.0D00) THEN
                LTLxT2 = DABS(RVCxALPH*RVCxZF).LT.(RVLxEPS*RVCxCZ)
                IF (LTLxT2) THEN
                    RVCxALPH = 0.0
                ELSE
                    IF (IFSxG.EQ.3) GO TO 10272
                    GO TO 10269
                ENDIF
            ENDIF
      C
      C     From here on, the code follows the logic outlined in the
      C        Radio Science paper referred to in the documentation.
      C
      10267 RTLxF3 = IVCxSZ*RVCxZF / DSQRT(RVCxCZ)
            GO TO 10293
      10269 RTLxR2 = RVCxCZ - RVCxALPH*RVCxZF
            IF (RTLxR2.GT.0.0D00) THEN
                RTLxF3 = DSQRT(RTLxR2)
            ELSE
                IFSxG = 3
                RVCxZF = RVCxCZ / RVCxALPH
                RTLxF3 = 0.0D00
                GO TO 10265
```

Line# Source Line

```
                ENDIF
      10272     RTLxF6 = DSQRT(RVCxCZ)
                RTLxF3 = RTLxF6 - RTLxF3
                RTLxF3 = 2.0D00*IVCxSZ*RTLxF3 / RVCxALPH
                GO TO 10293
      10280     RTLxR2 = RVCxBETA*RVCxZF*RVCxZF - RVCxALPH*RVCxZF - RVCxCZ
                IF (RTLxR2.GT.0.0D00) THEN
                    RTLxF4 = DSQRT(RTLxR2)
                ELSE
                    IFSxG = 3
                    RVCxZF = (RVCxALPH - IVCxSZ*DSQRT(RVCxALPH
            #*RVCxALPH - 4.0D00*RVCxBETA*RVCxCZ))/(2.0D00*RVCxBETA)
                    RTLxF4 = 0.0D00
                    GO TO 10263
                ENDIF
      10282     RTLxB1 = DSQRT(ABS(RVCxBETA))
                IF (RVCxBETA.LT.0.0) THEN
                    RTLxF6 = SQRT(RVCxALPH*RVCxALPH - 4.0*RVCxBETA*RVCxCZ)
                    IF (IFSxG.EQ.3) THEN
                        RTLxF4 = INTSIGN(-RVCxALPH)
                    ELSE
                        RTLxF4 = (2.0*RVCxBETA*RVCxZF - RVCxALPH)/RTLxF6
                    ENDIF
                    IF (DABS(-RVCxALPH/RTLxF6).GT.1.0) THEN
                        RTLxTEMP = DASIN(DFLOAT(INTSIGN(-RVCxALPH/RTLxF6)))
                    ELSE
                        RTLxTEMP = DASIN(-RVCxALPH/RTLxF6)
                    ENDIF
                    RTLxF3 = RTLxTEMP - DASIN(RTLxF4)
                ELSE
                    RTLxA2 = RVCxALPH / (2.0D00*RTLxB1)
                    RTLxF5 = DSQRT(RVCxCZ)
                    RTLxF6 = -RTLxA2 + RTLxF5
                    RTLxG3 = (RTLxB1*RVCxZF - RTLxA2 + RTLxF4) / RTLxF6
                    RTLxF3 = DLOG(RTLxG3)
                ENDIF
                RTLxF3 = IVCxSZ * RTLxF3 / RTLxB1
      10293     RTLxC4 = DSQRT(RVCxCX)
                RVSxGPI = RTLxF3
                IF (IFSxG.GT.5) GO TO 10352
                IF (RVCxET1.EQ.0.0D00) THEN
                    GO TO 10320
                ELSE
                    LTLxT3 = (DSQRT(RVCxCX)*DABS(RVCxET1*RVCxZF))
            #.LT.(RVLxEPS*RVCxCX*DSQRT(RVCxCZ))
                    IF (LTLxT3) THEN
                        RVCxET1 = 0.0D00
                        GO TO 10320
                    ENDIF
                ENDIF
```

```
Line#  Source Line

               IF (IVCxSX.EQ.0) IVCxSX = -INTSIGN(RVCxET1)
               RTLxC5 = RTLxC4 - RVCxET1*RTLxF3/(2.0D00*IVCxSX)
               IF (RTLxC5.GT.0.0D00) GO TO 10320
               RVCxXF = RVCxCX / RVCxET1
               IF (DABS(RVCxET1*RVCxXF).LT.(RVLxEPS*RVCxCX)) THEN
                   RVCxET1 = 0.0D00
                   GO TO 10320
               ENDIF
               IFSxG = 4
               RTLxF1 = 2.0D00*RTLxC4*IVCxSX / RVCxET1
               RVSxGPI = RTLxF1
               IF (RVCxBETA.EQ.0.0D00) THEN
                   IF (RVCxALPH.EQ.0.0D00) THEN
                       RVCxZF = RTLxF1*DSQRT(RVCxCZ)/IVCxSZ
                   ELSE
                       RVCxZF = (RVCxCZ - (RTLxF6 - RVCxALPH*RTLxF1/
            *(2.0D00*IVCxSZ))**2.0D00) / RVCxALPH
                   ENDIF
               ELSE
                   IF (RVCxBETA.LT.0.0) THEN
                       IF (DABS(-RVCxALPH/RTLxF6).GT.1.0) THEN
                          RTLxTEMP = DASIN(DFLOAT(INTSIGN(-RVCxALPH/RTLxF6,
                       ELSE
                          RTLxTEMP = DASIN(-RVCxALPH/RTLxF6)
                       ENDIF
                       RTLxF4 = RTLxTEMP - RTLxB1*RTLxF1*IVCxSZ
                       RTLxF4 = DSIN(RTLxF4)
                       RVCxZF = (RVCxALPH + RTLxF6*RTLxF4)/(2.0*RVCxBETA
                   ELSE
                       RTLxG1 = RTLxF6*DEXP(RTLxB1*RTLxF1/IVCxSZ) + RTLxA2
                       RVCxZF = (RTLxG1*RTLxG1 - RVCxCZ)/(2.0D00*RTLxB1*RTLxG
            *- RVCxALPH)
                   ENDIF
               ENDIF
               IF (RVCxET2.EQ.0.0D00) THEN
                   GO TO 10318
               ELSE
                   LTLxT4 = (DSQRT(RVCxCY)*DABS(RVCxET2*RVCxZF))
            #.LT.(RVLxEPS*RVCxCY*DSQRT(RVCxCZ))
                   IF (LTLxT4) THEN
                       RVCxET2 = 0.0D00
                       GO TO 10318
                   ENDIF
               ENDIF
               RTLxC6 = DSQRT(RVCxCY)
               RTLxC7 = RTLxC6 - RVCxET2*RTLxF1/(2.0D00*IVCxSY)
               IF (RTLxC7.LE.0.0D00) GO TO 10320
               RVCxYF = (RVCxCY - RTLxC7*RTLxC7)/RVCxET2
               GO TO 10341
       10318   RVCxYF = RTLxF1 * DSQRT(RVCxCY) / IVCxSY
```

Line# Source Line

```
            GO TO 10341
10320       RTLxC6 = DSQRT(RVCxCY)
            IF (RVCxET2.EQ.0.0D00) THEN
                GO TO 10341
            ELSE
                LTLxT4 = (DSQRT(RVCxCY)*DABS(RVCxET2*RVCxZF))
     #.LT.(RVLxEPS*RVCxCY*DSQRT(RVCxCZ))
                IF (LTLxT4) THEN
                    RVCxET2 = 0.0D00
                    GO TO 10341
                ENDIF
            ENDIF
            IF (IVCxSY.EQ.0) IVCxSY = INTSIGN(-RVCxET2)
            RTLxC7 = RTLxC6 - RVCxET2*RTLxF3/(2.0D00*IVCxSY)
            IF (RTLxC7.GT.0.0D00) GO TO 10341
            RVCxYF = RVCxCY / RVCxET2
            IF (DABS(RVCxET2*RVCxYF).LT.(RVLxEPS*RVCxCY)) THEN
                RVCxET2 = 0.0D00
                GO TO 10341
            ENDIF
            IFSxG = 5
            RTLxF2 = 2.0D00 * RTLxC6 * IVCxSY / RVCxET2
            RVSxGPI = RTLxF2
            IF (RVCxBETA.EQ.0.0D00) THEN
                IF (RVCxALPH.EQ.0.0D00) THEN
                    RVCxZF = RTLxF2 * DSQRT(RVCxCZ) / IVCxSZ
                ELSE
                    RVCxZF = (RVCxCZ - (RTLxF6 - RVCxALPH*RTLxF2/(
     *2.0D00*IVCxSZ))**2.0D00) / RVCxALPH
                ENDIF
            ELSE
                IF (RVCxBETA.LT.0.0) THEN
                    IF (DABS(-RVCxALPH/RTLxF6).GT.1.0) THEN
                        RTLxTEMP = DASIN(DFLOAT(INTSIGN(-RVCxALPH/RTLxF6
                    ELSE
                        RTLxTEMP = DASIN(-RVCxALPH/RTLxF6)
                    ENDIF
                    RTLxF4 = RTLxTEMP - RTLxB1*RTLxF2*IVCxSZ
                    RTLxF4 = DSIN(RTLxF4)
                    RVCxZF = (RVCxALPH + RTLxF6*RTLxF4)/(2.0*RVCxBETA
                ELSE
                    RTLxG2 = RTLxF6*DEXP(RTLxB1*RTLxF2/IVCxSZ) + RTLxA2
                    RVCxZF = (RTLxG2*RTLxG2 - RVCxCZ)/(2.0D00*RTLxB1*RTLxG2
     * - RVCxALPH)
                ENDIF
            ENDIF
            IF (RVCxET1.EQ.0.0D00) THEN
                RVCxXF = RTLxF2 * DSQRT(RVCxCX) / IVCxSX
            ELSE
                RTLxC5 = RTLxC4 - RVCxET1*RTLxF2/(2.0D00*IVCxSX)
```

```
Line#  Source Line

                 RVCxXF = (RVCxCX - RTLxC5*RTLxC5)/RVCxET1
             ENDIF
   10341     IF (IFSxG.GT.3) RETURN
             IF (RVCxET1.NE.0.0D00) THEN
                 RVCxXF = (RVCxCX - RTLxC5*RTLxC5)/RVCxET1
             ELSE IF (RVCxCX.NE.0.0D00) THEN
                 RVCxXF = RTLxC4 * IVCxSX * RTLxF3
             ENDIF
             IF (RVCxET2.NE.0.0D00) THEN
                 RVCxYF = (RVCxCY - RTLxC7*RTLxC7)/RVCxET2
                 RETURN
             ENDIF
             IF (RVCxCY.EQ.0.0D00) RETURN
             RVCxYF = RTLxC6 * IVCxSY * RTLxF3
   10352     RETURN
             END
```

ENDPT Local Symbols

Name	Class	Type	Size
RVSXGPI	param		
RVSXS4.	param		
RVSXHZ.	param		
RVSXH0.	param		
IFSXG	param		
RTLXTEMP.	local	REAL*8	8
RTLXC4.	local	REAL*8	8
RTLXG1.	local	REAL*8	8
RTLXC5.	local	REAL*8	8
RTLXG2.	local	REAL*8	8
RTLXC6.	local	REAL*8	8
RTLXC7.	local	REAL*8	8
RTLXG3.	local	REAL*8	8
RTLXF4.	local	REAL*8	8
RTLXF5.	local	REAL*8	8
RTLXF6.	local	REAL*8	8
LTLXT1.	local	LOGICAL*4	4
LTLXT2.	local	LOGICAL*4	4
LTLXT3.	local	LOGICAL*4	4
LTLXT4.	local	LOGICAL*4	4
RTLXR2.	local	REAL*8	8
RVLXEPS	local	REAL*8	8
RTLXA2.	local	REAL*8	8
RTLXB1.	local	REAL*8	8
IVCXSX.	MORE	INTEGER*4	4
IVCXSY.	MORE	INTEGER*4	4
IVCXSZ.	MORE	INTEGER*4	4
RVCXCX.	MORE	REAL*8	8

ENDPT Local Symbols

Name	Class	Type	Size
RVCXCY.	MORE	REAL*8	8
RVCXCZ.	MORE	REAL*8	3
RVCXLATO.	OTHER	REAL*8	8
RVCXLONO.	OTHER	REAL*8	8
RVCXHB.	OTHER	REAL*8	8
RVCXXF.	END	REAL*8	8
RVCXYF.	END	REAL*8	8
RVCXZF.	END	REAL*8	8
RVCXH5.	END	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RTLXF3.	TEMP4	REAL*8	8
RTLXF1.	TEMP4	REAL*8	8
RTLXF2.	TEMP4	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RPCXDTR	PRAM	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RPCXHTP	PRAM	REAL*8	8

```
Line#  Source Line

         SUBROUTINE PHSPL(IFSxG, RVSxPPI)
C
C-----------------------------------------------------------------
C
C        PHSPL  -- SUBROUTINE TO CALCULATE THE PHASE PATH LENGTH
C                  INCREMENT FOR THE CURRENT RAY PATH INCREMENT.
C
C
C        CALLED BY:       RAYSUB
C
C        CALLS:           PHSPL2, PHSPL3
C
C-----------------------------------------------------------------
C
C        AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
C
C        DATE:            09/30/87
C
C        VERSION:         1.0
C
C-----------------------------------------------------------------
C
C        REVISED:         09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C        USES:            RVCxALPH \
C                         RVCxBETA |____   To decide on which PPI calc.
C                         RVCxET1  |          to use.
C                         RVCxET2  /
C
C                         IVCxSX,Y,Z  \
C                         RVCxCX,Y,Z  |-- To calc. PPI in regions of
C                         RVCxX,Y,ZF  /      constant plasma frequency.
C
C                         IFSxG            A flag variable.
C
C        To either pass off the phase path increment length calculation
C          to PHSPL2,3 or to do it locally for the simplest case.  The
C          phase path calculations are done by considering the (mu)*ds
C          integral, where (mu) is the index of refraction.  (See the
C          Radio Science paper referred to in the documentation, eq. 7.
C
C
C        RETURNS:         RVSxPPI        The increment of phase path
C                                          length.
C
C
C-----------------------------------------------------------------
C
```

Line#   Source Line

```
C
        INTEGER IFSxG, IVCxSX, IVCxSY, IVCxSZ
C
        REAL*8 RVSxPPI, RVCxF1, RVCxF2, RVCxF3, RVCxBETA, RVCxXF
        REAL*8 RVCxYF, RVCxZF, RVCxCX, RVCxCY, RVCxCZ, RVCxALPH
        REAL*8 RVSxPPIX, RVSxPPIY, RVSxPPIZ, RVCxET1, RVCxET2
        REAL*8 RTCxA
C
        COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RTCxA
        COMMON /MORE/ IVCxSX,IVCxSY,IVCxSZ,RVCxCX,RVCxCY,RVCxCZ
        COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2
        COMMON /TEMP4/ RVCxF1, RVCxF2, RVCxF3
C
C       IFSxG = 4  to flag an x-reflection & = 5 to flag a
C         y-reflection.
C
        IF (IFSxG.EQ.4) RVCxF3 = RVCxF1
        IF (IFSxG.EQ.5) RVCxF3 = RVCxF2
        IF (RVCxBETA.NE.0.0D00) THEN
C
C       Call PHSPL3 to calculate the Z phase path increment when
C          beta is not equal to zero.
C
            CALL PHSPL3(RVCxZF,RVCxF3,IVCxSZ,RVCxCZ,RVCxALPH,RVCxBETA
     #,IFSxG,RVSxPPIZ)
        ELSE IF (RVCxALPH.EQ.0.0D00) THEN
            RVSxPPIZ = IVCxSZ * RVCxZF * DSQRT(RVCxCZ)
        ELSE
C
C       Call PHSPL2 to calculate phase path increment for a direction
C          in which the plasma frequency has a linear dependence.
C
            CALL PHSPL2(RVCxZF,IVCxSZ,RVCxCZ,RVCxALPH,RVSxPPIZ)
        ENDIF
        IF (RVCxET1.EQ.0.0D00) THEN
            RVSxPPIX = IVCxSX * RVCxXF * DSQRT(RVCxCX)
        ELSE
            CALL PHSPL2(RVCxXF,IVCxSX,RVCxCX,RVCxET1,RVSxPPIX)
        ENDIF
        IF (RVCxET2.EQ.0.0D00) THEN
            RVSxPPIY = IVCxSY * RVCxYF * DSQRT(RVCxCY)
        ELSE
            CALL PHSPL2(RVCxYF,IVCxSY,RVCxCY,RVCxET2,RVSxPPIY)
        ENDIF
        RVSxPPI = RVSxPPIX + RVSxPPIY + RVSxPPIZ
        RETURN
        END
```

PHSPL Local Symbols

Name	Class	Type	Size
RVSXPPI	param		
IFSXG	param		
RVSXPPIX.	local	REAL*8	8
RVSXPPIY.	local	REAL*8	8
RVSXPPIZ.	local	REAL*8	8
RVCXCY.	MORE	REAL*8	8
RVCXCZ.	MORE	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RTCXA	END	REAL*8	8
IVCXSX.	MORE	INTEGER*4	4
IVCXSY.	MORE	INTEGER*4	4
IVCXSZ.	MORE	INTEGER*4	4
RVCXF1.	TEMP4	REAL*8	8
RVCXF2.	TEMP4	REAL*8	8
RVCXF3.	TEMP4	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXXF.	END	REAL*8	8
RVCXYF.	END	REAL*8	8
RVCXZF.	END	REAL*8	8
RVCXCX.	MORE	REAL*8	8

Line# Source Line

```
      SUBROUTINE PHSPL3(RVSxZ5,RVSxF3,IVSxSZ,RVSxCZ,RVSxALPH
     #,RVSxBETA,IFSxG,RVSxPPIZ)
C
C-----------------------------------------------------------------
C
C       PHSPL3  -- SUBROUTINE TO CALCULATE THE Z INCREMENT OF
C                      PHASE PATH LENGTH FOR BETA EQUALS ZERO.
C
C
C       CALLED BY:      PHSPL
C
C-----------------------------------------------------------------
C
C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C       DATE:           09/30/87
C
C       VERSION:        1.0
C
C-----------------------------------------------------------------
C
C       REVISED:        09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C       USES:           RVSxZ5          SEZ z value at end of RPI.
C
C                       RVSxF3          Complete group path length
C                                          increment.
C
C                       IVSxSZ   \
C                       RVSxCZ   |___   Propagation parameters.
C                       RVSxALPH |
C                       RVSxBETA /
C
C                       IFSxG           A flag variable.
C
C
C       To calculate the increment of phase path length in the
C         z-direction when the z dependence of the plasma freq.
C         is quadratic (i.e. beta.NE.0).
C
C
C       RETURNS:        RVSxPPIZ        The z increment of phase
C                                          path length.
C
C-----------------------------------------------------------------
C
      INTEGER IVSxSZ,IFSxG
C
```

Line#   Source Line

```
          REAL*8 RVSxZ5, RVSxF3, RVSxCZ, RVSxALPH, RVSxBETA
          REAL*8 RVSxPPIZ, RVLxR2, RVLxDEL
C
          IF (IFSxG.EQ.3) THEN
             RVLxR2 = 0.0D00
          ELSE
             RVLxR2 = DSQRT(RVSxCZ - RVSxALPH*RVSxZ5 + RVSxBETA*
     #RVSxZ5*RVSxZ5)
          ENDIF
          RVLxDEL = 4.0D00 * RVSxBETA * RVSxCZ - RVSxALPH * RVSxALPH
          RVSxPPIZ = (2.0D00 * RVSxBETA * RVSxZ5 - RVSxALPH) * RVLxR2
          RVSxPPIZ = RVSxPPIZ + RVSxALPH * DSQRT(RVSxCZ)
          RVSxPPIZ = (IVSxSZ * RVSxPPIZ + RVLxDEL * RVSxF3/2.0D00)
          RVSxPPIZ = RVSxPPIZ / (4.0D00 * RVSxBETA)
          RETURN
          END
```

PHSPL3 Local Symbols

Name	Class	Type	Size
RVSXPPIZ.	param		
IFSXG	param		
RVSXBETA.	param		
RVSXALPH.	param		
RVSXCZ.	param		
IVSXSZ.	param		
RVSXF3.	param		
RVSXZ5.	param		
RVLXR2.	local	REAL*8	8
RVLXDEL	local	REAL*8	8

Line# Source Line

```
      SUBROUTINE PHSPL2(RVSxW, IVSxS, RVSxC, RVSxN, RVSxPPIW)
C
C-----------------------------------------------------------------
C
C     PHSPL2  -- SUBROUTINE TO CALCULATE THE PHASE PATH LENGTH
C                INCREMENT FOR LINEAR PLASMA FREQUENCY PATHS.
C
C
C     CALLED BY:      PHSPL
C
C-----------------------------------------------------------------
C
C     AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C     DATE:           09/30/87
C
C     VERSION:        1.0
C
C-----------------------------------------------------------------
C
C     REVISED:        09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C     USES:           RVSxW        The generalized SEZ coord.
C                                 value at the end of the RPI.
C
C                     IVSxS   \
C                     RVSxC   |--  The generalized propagation
C                     RVSxN   /    parameters.
C
C
C     To calculate the phase path length increment for the
C       particular coordinate direction of interest.  The PPI
C       is calculated for a plasma frequency that is linearly
C       dependent on the coordinate involved.  This translates
C       to having N = Nx or Ny, or if B = 0, N = A.  The same
C       routine can then be used in any of the coordinate
C       directions.
C
C
C     RETURNS:        RVSxPPIW     The phase path increment for
C                                 the particular coordinate dir
C
C-----------------------------------------------------------------
C
      INTEGER IVSxS
C
      REAL*8 RVSxW, RVSxC, RVSxN, RVSxPPIW, RVLxR, RVLxR2
C
```

```
Line#  Source Line

           RVLxR = RVSxC - RVSxN * RVSxW
           IF (RVLxR.GT.0.0D00) THEN
               RVLxR2 = -RVLxR * DSQRT(RVLxR)
           ELSE
               RVLxR2 = 0.0D00
           ENDIF
           RVSxPPIW = 2.0D00 * IVSxS * (RVLxR2 + RVSxC * DSQRT(RVSxC))
           RVSxPPIW = RVSxPPIW / (3.0D00 * RVSxN)
           RETURN
           END
```

PHSPL2 Local Symbols

Name	Class	Type	Size
RVSXPPIW.	param		
RVSXN	param		
RVSXC	param		
IVSXS	param		
RVSXW	param		
RVLXR2.	local	REAL*8	8
RVLXR	local	REAL*8	8

Global Symbols

Name	Class	Type	Size
ACCFSP.	extern	***	***
END	common	***	32
ENDPT	FSUBRT	***	***
INTSIGN	extern	INTEGER*4	***
IONO1	common	***	32
MORE.	common	***	36
OTHER	common	***	24
PHSPL	FSUBRT	***	***
PHSPL2.	FSUBRT	***	***
PHSPL3.	FSUBRT	***	***
PRAM.	common	***	32
TEMP4	common	***	24

Code size = 155e (5470)
Data size = 0044 (68)

Line# Source Line

```
      SUBROUTINE TILTS(RVSxDXHB, RVSxDYHB, IVSxJ)
C
C-----------------------------------------------------------------
C
C  TILTS -- SUBROUTINE TO CALCULATE THE IONOSPHERIC BOUNDARY TILTS
C
C        CALLED BY: RAYSUB
C
C-----------------------------------------------------------------
C
C        AUTHOR:             MICHAEL H. REILLY & ERIC L. STROBEL
C
C        DATE:               09/30/87
C
C        VERSION:            2.0
C
C-----------------------------------------------------------------
C
C        REVISED:            07/25/86 -- INITIAL REVISION.  TRANSLATED
C                            FROM TEKTRONIX BASIC TO VAX FORTRAN BY
C                            ERIC L. STROBEL.
C
C                            07/30/86 -- V1.1.  Change over to use of
C                            REAL*8 precision in the calculations.
C
C                            09/30/87 -- V2.0.  Extensively modified
C                            to accommodate the use of RADAR-C.  This
C                            routine now serves as a dispatcher for
C                            the real processing.
C
C-----------------------------------------------------------------
C
C       USES:                IVSxJ            A boundary index.
C
C                            IVCxCASE         Describes the compo. of
C                                            profiles used.
C
C                            RVCxHBND         Height of the upcoming
C                                            boundary.
C
C                            IFCxSN           A flag parameter needed in
C                                            one of the cases.
C
C        To decide which of the tilt calculation routines is to be
C          used, if at all.  The J-index values have different
C          meanings depending on which CASE applies.
C
C        RETURNS:            RVSxDXHB         (d/dx)Boundary Height
C                            RVSxDYHB         (d/dy)Boundary Height
C
```

Line#   Source Line

```
C-------------------------------------------------------------------
C
         INTEGER IVSxJ, IVCxCASE, IFCxSN
C
         REAL*8 RVSxDXHB, RVSxDYHB, RPCxPI, RPCxDR, RPCxRE
         REAL*8 RVCxV1, RVCxV2, RPCxHTP, RVCxXX, RVCxHBND
C
         COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
         COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
         COMMON /RAID/ IFCxSN
C
         RVSxDXHB = 0.0D00
         RVSxDYHB = 0.0D00
         IF (IVSxJ.LT.5) RETURN
         IF (RVCxHBND.GE.RPCxHTP) RETURN
C
C        Begin calling the routines for the boundary tilt calculation,
C          based upon the CASE and J values.
C
         IF (IVCxCASE.EQ.1) THEN
             IF (IVSxJ.EQ.5) CALL HUTLT(RVSxDXHB, RVSxDYHB)
             IF (IVSxJ.EQ.6) CALL H5TLT(RVSxDXHB, RVSxDYHB)
         ELSE IF (IVCxCASE.EQ.2) THEN
             IF (IVSxJ.EQ.5) CALL H1PTLT(1,RVSxDXHB, RVSxDYHB)
             IF (IVSxJ.EQ.6) CALL H4TLT(IFCxSN, RVSxDXHB, RVSxDYHB)
             IF (IVSxJ.EQ.7) CALL H5TLT(RVSxDXHB, RVSxDYHB)
         ELSE IF (IVCxCASE.EQ.3) THEN
             IF (IVSxJ.EQ.5) CALL H1LTLT(RVSxDXHB, RVSxDYHB)
             IF (IVSxJ.EQ.6) CALL H2TLT(RVSxDXHB, RVSxDYHB)
             IF (IVSxJ.EQ.7) CALL H5TLT(RVSxDXHB, RVSxDYHB)
         ELSE IF (IVCxCASE.EQ.4) THEN
             IF (IVSxJ.EQ.5) CALL HUTLT(RVSxDXHB, RVSxDYHB)
             IF (IVSxJ.EQ.6) CALL H4TLT(-1, RVSxDXHB, RVSxDYHB)
             IF (IVSxJ.EQ.7) CALL H4TLT(1, RVSxDXHB, RVSxDYHB)
             IF (IVSxJ.EQ.8) CALL H5TLT(RVSxDXHB, RVSxDYHB)
         ELSE IF (IVCxCASE.EQ.5) THEN
             IF (IVSxJ.EQ.5) CALL H1PTLT(1,RVSxDXHB,RVSxDYHB)
             IF (IVSxJ.EQ.6) CALL H1PTLT(-1,RVSxDXHB,RVSxDYHB)
             IF (IVSxJ.EQ.7) CALL HUTLT(RVSxDXHB,RVSxDYHB)
             IF (IVSxJ.EQ.8) CALL H5TLT(RVSxDXHB,RVSxDYHB)
         ENDIF
         RETURN
         END
```

TILTS Local Symbols

Name	Class	Type	Size
IVSXJ	param		

TILTS Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
IVCXCASE.	IONO3	INTEGER*4	4
IFCXSN.	RAID	INTEGER*4	4
RPCXPI.	PRAM	REAL*8	8
RPCXDR.	PRAM	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RVCXXX.	IONO3	REAL*8	8
RVCXHBND.	IONO3	REAL*8	8

Line# Source Line

```
      SUBROUTINE HUTLT(RVSxDXHB, RVSxDYHB)
C
C-----------------------------------------------------------------
C
C     HUTLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
C                    VALLEY-F2 BOUNDARY
C
C     CALLED BY:      TILTS
C
C-----------------------------------------------------------------
C
C     AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C     DATE:           09/30/87
C
C     VERSION:        1.0
C
C-----------------------------------------------------------------
C
C     REVISED:        09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C     USES:           The parameters from the IONO1,2,3 blocks.
C
C
C     To calculate the rate of change of the boundary in the
C       locally horizontal directions.  The boundary here is
C       the boundary between the linear valley profile and the
C       F2 parabolic profile (for those cases where the inter-
C       section is visible on the final profile).  For further
C       details, see the RADAR-C report referred to in the
C       documentation.
C
C
C     RETURNS:        RVSxDXHB          (d/dx) Height of Boundary.
C
C                     RVSxDYHB          (d/dy) Height of Boundary.
C
C-----------------------------------------------------------------
C
      INTEGER IVCxCASE
C
      REAL*8 RVSxDXHB, RVSxDYHB, RTLxD, RVCxHB(3), RVCxHBND
      REAL*8 RVLxHUY2, RVLxHUXE, RVCxFB(3), RVLxHUX2, RVLxHUH2
      REAL*8 RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxE6(3)
      REAL*8 RVCxA5(3), RVCxA6(3), RVCxV1, RVCxV2, RVCxXX
      REAL*8 RVCxALPH, RVCxBETA, RVCxET1, RVCxET2
C
      COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
```

```
Line#  Source Line

            COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
            COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
      C
            RTLxD = (RVCxHB(2) - RVCxHBND) / RVCxHB(3)
            RVLxHUY2 = -RTLxD
            RVLxHUXE = 0.9604D00 * RVCxHB(3) / (2.0D00 * RTLxD * RVCxFB(3))
            RVLxHUX2 = -RVLxHUXE * RVCxFB(1) / RVCxFB(3)
            RVLxHUH2 = 1.0D00
            RVSxDXHB = RVLxHUXE*RVCxB5(1) + RVLxHUX2*RVCxB5(3)
            RVSxDXHB = RVSxDXHB + RVLxHUH2*RVCxB6(2) + RVLxHUY2*RVCxB6(3)
            RVSxDXHB = -RVSxDXHB / RVCxV1
            RVSxDYHB = RVLxHUXE*RVCxE5(1) + RVLxHUX2*RVCxE5(3)
            RVSxDYHB = RVSxDYHB + RVLxHUH2*RVCxE6(2) + RVLxHUY2*RVCxE6(3)
            RVSxDYHB = RVSxDYHB / RVCxV2
            RETURN
            END
```

HUTLT Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
RVLXHUXE.	local	REAL*8	8
RTLXD	local	REAL*8	8
RVLXHUH2.	local	REAL*8	8
RVLXHUX2.	local	REAL*8	8
RVLXHUY2.	local	REAL*8	8
IVCXCASE.	IONO3	INTEGER*4	4
RVCXHB.	IONO1	REAL*8	24
RVCXHBND.	IONO3	REAL*8	8
RVCXFB.	IONO1	REAL*8	24
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXXX.	IONO3	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8

Line# Source Line

```
          SUBROUTINE H5TLT(RVSxDXHB, RVSxDYHB)
C
C-----------------------------------------------------------------
C
C         H5TLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
C                        F2-TOPSIDE BOUNDARY
C
C         CALLED BY:       TILTS
C
C-----------------------------------------------------------------
C
C         AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
C
C         DATE:            09/30/87
C
C         VERSION:         1.0
C
C-----------------------------------------------------------------
C
C         REVISED:         09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C         USES:            The parameters from the IONO2,3 blocks.
C
C
C         To calculate the rate of change of the boundary in the
C           locally horizontal directions.  The boundary here is
C           the boundary between the parabolic F2 profile and the
C           Bent topside profile.  For further details, see the
C           RADAR-C report referred to in the documentation.
C
C
C         RETURNS:         RVSxDXHB          (d/dx) Height of Boundary.
C
C                          RVSxDYHB          (d/dy) Height of Boundary.
C
C-----------------------------------------------------------------
C
          INTEGER ITCxA
C
          REAL*8 RVSxDXHB, RVSxDYHB, RVLxH5Y2, RVLxH5H2
          REAL*8 RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxE6(3)
          REAL*8 RVCxA5(3), RVCxA6(3), RVCxV1, RVCxV2
          REAL*8 RTCxA, RTCxB
C
          COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
          COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
C
          RVLxH5H2 = 1.0D00
```

```
Line#   Source Line

            RVLxH5Y2 = 0.25D00
            RVSxDXHB = -(RVLxH5H2*RVCxB6(2) + RVLxH5Y2*RVCxB6(3)) / RVCxV1
            RVSxDYHB = (RVLxH5H2*RVCxE6(2) + RVLxH5Y2*RVCxE6(3)) / RVCxV2
            RETURN
            END
```

H5TLT Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
RVLXH5H2.	local	REAL*8	8
RVLXH5Y2.	local	REAL*8	8
ITCXA	IONO3	INTEGER*4	4
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8

Line# Source Line

```
      SUBROUTINE H1PTLT(IVSxSGN,RVSxDXHB, RVSxDYHB)
C
C-----------------------------------------------------------------------
C
C     H1PTLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
C                  VALLEY-F1 (PARABOLIC) BOUNDARY
C
C     CALLED BY:     TILTS
C
C     CALLS:         DH1PDP
C
C-----------------------------------------------------------------------
C
C     AUTHOR:        MICHAEL H. REILLY & ERIC L. STROBEL
C
C     DATE:          09/30/87
C
C     VERSION:       1.0
C
C-----------------------------------------------------------------------
C
C     REVISED:       09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------------
C
C     USES:          The parameters from the COMMON blocks.
C
C
C     To calculate the rate of change of the boundary in the
C       locally horizontal directions.  The boundary here is
C       the boundary between the linear valley profile and the
C       parabolic F1 profile.  The routine DH1PDP is just
C       d (H1P) / d (parameter).  For further details, see the
C       RADAR-C report referred to in the documentation.
C
C
C     RETURNS:       RVSxDXHB         (d/dx) Height of Boundary.
C
C                    RVSxDYHB         (d/dy) Height of Boundary.
C
C-----------------------------------------------------------------------
C
      INTEGER IVSxSGN, ITCxA
C
      REAL*8 RVSxDXHB, RVSxDYHB, RVLxD1, RVCxA1, RVCxB1, RVCxC1
      REAL*8 RVLxBB, RVCxB0, RVCxHU, RVCxHL, RVCxHB(3), RVCxFB(3
      REAL*8 RVLxHUXE, RVLxB1XE, RVLxC1XE, RVCxA0, RVLxHPXE
      REAL*8 RVLxA1X1, RVLxB1X1, RVLxHPX1, RVLxA1H1, RVLxB1H1
      REAL*8 RVLxHPH1, RVLxHUX2, RVLxB1X2, RVLxC1X2, RVLxHPX2
      REAL*8 RVLxB1H2, RVLxC1H2, RVLxHPH2, RVLxHUY2, RVLxB1Y2
```

Line# Source Line

```
      REAL*8 RVLxC1Y2, RVLxHPY2, RVCxB5(3), RVCxB6(3), RVCxE5(3)
      REAL*8 RVCxE6(3), RVCxV1, RVCxV2, RVCxALPH, RVCxBETA
      REAL*8 RVCxET1, RVCxET2, RVCxA5(3), RVCxA6(3)
      REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU
      REAL*8 RTCxA, RTCxB, RTCxC, RTCxD
C
      COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
      COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
      COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
      COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
      COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
      COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RTCxC, RTCxD
C
      RVLxD1 = DSQRT(RVCxB1*RVCxB1 - 4.0D00*RVCxA1*RVCxC1)
      RVLxBB = RVCxB0 / (RVCxHU - RVCxHL)
      RVLxHUXE = (0.98D00 * RVCxHB(3))*(0.98D00 * RVCxHB(3))
      RVLxHUXE = RVLxHUXE /(2.0D00 * RVCxFB(3)*(RVCxHB(2)-RVCxHU))
      RVLxB1XE = RVCxB0 * (-1.0D00 + (RVCxFB(1)*RVLxHUXE)/
     #(RVCxHU - RVCxHL)) / RVCxFB(1)
      RVLxC1XE = RVCxA0/RVCxFB(1) + RVCxHL*RVLxBB*RVLxHUXE
      CALL DH1PDP(0.0D00, RVLxB1XE, RVLxC1XE, RVLxHPXE, RVLxD1,
     #IVSxSGN)
      RVLxA1X1 = 16.0D00 / (RVCxHB(1) * RVCxHB(1))
      RVLxB1X1 = 32.0D00 / RVCxHB(1)
      CALL DH1PDP(RVLxA1X1, RVLxB1X1, 15.0D00, RVLxHPX1, RVLxD1,
     #IVSxSGN)
      RVLxA1H1 = -32.0D00*RVCxFB(2)/(RVCxHB(1)*RVCxHB(1)*RVCxHB(1))
      RVLxB1H1 = RVLxA1H1 * RVCxHB(1)
      CALL DH1PDP(RVLxA1H1, RVLxB1H1, 0.0D00, RVLxHPH1, RVLxD1,
     #IVSxSGN)
      RVLxHUX2 = -RVLxHUXE * RVCxFB(1) / RVCxFB(3)
      RVLxB1X2 = RVLxBB * RVLxHUX2
      RVLxC1X2 = RVCxHL * RVLxB1X2
      CALL DH1PDP(0.0D00, RVLxB1X2, RVLxC1X2, RVLxHPX2, RVLxD1,
     #IVSxSGN)
      RVLxB1H2 = RVLxBB
      RVLxC1H2 = RVCxHL * RVLxBB
      CALL DH1PDP(0.0D00, RVLxB1H2, RVLxC1H2, RVLxHPH2, RVLxD1,
     #IVSxSGN)
      RVLxHUY2 = -(RVCxHB(2) - RVCxHU) / RVCxHB(3)
      RVLxB1Y2 = RVLxBB * RVLxHUY2
      RVLxC1Y2 = RVCxHL * RVLxB1Y2
      CALL DH1PDP(0.0D00, RVLxB1Y2, RVLxC1Y2, RVLxHPY2, RVLxD1,
     #IVSxSGN)
      RVSxDXHB = RVLxHPXE*RVCxB5(1) + RVLxHPX1*RVCxB5(2) +
     #RVLxHPX2*RVCxB5(3)
      RVSxDXHB = RVSxDXHB + RVLxHPH1*RVCxB6(1) + RVLxHPH2*RVCxB6(2)
     #+ RVLxHPY2*RVCxB6(3)
      RVSxDXHB = -RVSxDXHB / RVCxV1
      RVSxDYHB = RVLxHPXE*RVCxE5(1) + RVLxHPX1*RVCxE5(2) +
```

```
Line#  Source Line

       #RVLxHPX2*RVCxE5(3)
          RVSxDYHB = RVSxDYHB + RVLxHPH1*RVCxE6(1) + RVLxHPH2*RVCxE6(2)
       #+ RVLxHPY2*RVCxE6(3)
          RVSxDYHB = RVSxDYHB / RVCxV2
          RETURN
          END
```

H1PTLT Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
IVSXSGN	param		
RVLXHPXE.	local	REAL*8	8
RVLXD1.	local	REAL*8	8
RVLXHUXE.	local	REAL*8	8
RVLXA1X1.	local	REAL*8	8
RVLXB1X1.	local	REAL*8	8
RVLXB1X2.	local	REAL*8	8
RVLXB1Y2.	local	REAL*8	8
RVLXC1X2.	local	REAL*8	8
RVLXC1Y2.	local	REAL*8	8
RVLXBB.	local	REAL*8	8
RVLXB1XE.	local	REAL*8	8
RVLXHPH1.	local	REAL*8	8
RVLXC1XE.	local	REAL*8	8
RVLXHPH2.	local	REAL*8	8
RVLXHPX1.	local	REAL*8	8
RVLXHPX2.	local	REAL*8	8
RVLXHPY2.	local	REAL*8	8
RVLXHUX2.	local	REAL*8	8
RVLXHUY2.	local	REAL*8	8
RVLXA1H1.	local	REAL*8	8
RVLXB1H1.	local	REAL*8	8
RVLXB1H2.	local	REAL*8	8
RVLXC1H2.	local	REAL*8	8
RVCXA0.	VAR1	REAL*8	8
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXA5.	IONO2	REAL*8	24

H1PTLT Local Symbols

Name	Class	Type	Size
RVCXA6.	IONO2	REAL*8	24
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXK1.	TEMP1	REAL*8	8
RVCXXL.	VAR1	REAL*8	8
RVCXXU.	VAR1	REAL*8	8
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8
RTCXC	VAR3	REAL*8	8
RTCXD	VAR3	REAL*8	8
ITCXA	IONO3	INTEGER*4	4
RVCXA1.	VAR3	REAL*8	8
RVCXB1.	VAR3	REAL*8	8
RVCXC1.	VAR3	REAL*8	8
RVCXB0.	VAR1	REAL*8	8
RVCXHU.	VAR1	REAL*8	8
RVCXHL.	TEMP1	REAL*8	8
RVCXHB.	IONO1	REAL*8	24
RVCXFB.	IONO1	REAL*8	24

Line# Source Line

```
      SUBROUTINE DH1PDP(RVSxA1P, RVSxB1P, RVSxC1P, RVSxDD, RVSxD1,
     #IVSxSGN)
C
C-----------------------------------------------------------------
C
C     DH1PDP  -- SUBROUTINE TO CALCULATE THE DERIVATIVE OF THE
C                BOUNDARY HEIGHT W.R.T. A PARTICULAR PARAMETER.
C
C
C     CALLED BY:      H1PTLT
C
C-----------------------------------------------------------------
C
C     AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C     DATE:           09/30/87
C
C     VERSION:        1.0
C
C-----------------------------------------------------------------
C
C     REVISED:        09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C     USES:           RVSxA,B,C1P    The partials of the para-
C                                   bolic profile parameters w.r.t.
C                                   the parameter of interest.
C
C                     RVSxD1         The discriminant of the para-
C                                   bolic parameters.
C
C                     IVSxSGN        Determines which of the sol'ns.
C                                   of the quadratic eqn. applies.
C
C     To calculate the partial derivative of H1P with respect to
C       the chosen parameter.  For further details, see the
C       RADAR-C report referred to in the documentation.
C
C
C     RETURNS:        RVSxDD         The aforementioned partial
C                                   derivative.
C
C-----------------------------------------------------------------
C
      INTEGER IVSxSGN
C
      REAL*8 RVSxA1P, RVSxB1P, RVSxC1P, RVSxDD, RVSxD1, RVLxH1
      REAL*8 RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
C
```

```
Line#  Source Line

            COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
      C
            RVLxH1 = RVCxH1P
            RVSxDD = (-RVLxH1 + IVSxSGN*RVCxC1/RVSxD1) * RVSxA1P
            RVSxDD = RVSxDD + (1.0D00 - IVSxSGN*RVCxB1/RVSxD1)
         #*RVSxB1P/2.0D00
            RVSxDD = (RVSxDD + IVSxSGN*(RVCxA1/RVSxD1) * RVSxC1P)
         #/ RVCxA1
            RETURN
            END
```

DH1PDP Local Symbols

Name	Class	Type	Size
IVSXSGN	param		
RVSXD1.	param		
RVSXDD.	param		
RVSXC1P	param		
RVSXB1P	param		
RVSXA1P	param		
RVLXH1.	local	REAL*8	8
RVCXA1.	VAR3	REAL*8	8
RVCXB1.	VAR3	REAL*8	8
RVCXC1.	VAR3	REAL*8	8
RVCXH1P	VAR3	REAL*8	8
RVCXH2P	VAR3	REAL*8	8

Line# Source Line

```
      SUBROUTINE H4TLT(IVSxSN, RVSxDXHB, RVSxDYHB)
C
C-----------------------------------------------------------------
C
C     H4TLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
C                     PARABOLIC F1-F2 BOUNDARY
C
C     CALLED BY:      TILTS
C
C     CALLS:          DH4DP
C
C-----------------------------------------------------------------
C
C     AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C     DATE:           09/30/87
C
C     VERSION:        1.0
C
C-----------------------------------------------------------------
C
C     REVISED:        09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C     USES:           The parameters from the COMMON blocks.
C
C
C     To calculate the rate of change of the boundary in the
C       locally horizontal directions.  The boundary here is
C       the boundary between the parabolic F1 profile and the
C       parabolic F2 profile.  The routine DH4DP is just
C       d (H4) / d (parameter).  For further details, see the
C       RADAR-C report referred to in the documentation.
C
C
C     RETURNS:        RVSxDXHB          (d/dx) Height of Boundary.
C
C                     RVSxDYHB          (d/dy) Height of Boundary.
C
C-----------------------------------------------------------------
C
      INTEGER IVSxSN, ITCxA
C
      REAL*8 RVSxDXHB, RVSxDYHB, RVLxD2, RVCxA2, RVCxB2, RVCxC2
      REAL*8 RVCxHT4, RVCxHB(3), RVCxFB(3), RVLxBB1, RVCxHT3
      REAL*8 RVLxBB2, RVLxA2X1, RVLxB2X1, RVLxH4X1, RVLxA2X2
      REAL*8 RVLxB2X2, RVLxC2X2, RVLxH4X2, RVLxA2H1, RVLxB2H1
      REAL*8 RVLxH4H1, RVLxB2H2, RVLxC2H2, RVLxH4H2, RVLxA2Y2
      REAL*8 RVLxB2Y2, RVLxC2Y2, RVLxH4Y2, RVCxB5(3), RVCxB6(3)
```

```
Line#  Source Line

          REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxALPH
          REAL*8 RVCxBETA, RVCxET1, RVCxET2, RVCxA5(3), RVCxA6(3)
          REAL*8 RTCxA, RTCxB
C
          COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
          COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
          COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
          COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4
C
          RVLxD2 = DSQRT(RVCxB2*RVCxB2 - 4.0D00*RVCxA2*RVCxC2)*IVSxSN
     #*INTSIGN(RVCxB2)
          RVCxHT4 = -(RVCxB2 + RVLxD2) / (2.0D00 * RVCxA2)
          RVLxBB1 = 16.0D00 / (RVCxHB(1) * RVCxHB(1))
          RVLxBB2 = 1.0D00 / (RVCxHB(3) * RVCxHB(3))
          RVLxA2X1 = -RVLxBB1
          RVLxB2X1 = 32.0D00 / RVCxHB(1)
          CALL DH4DP(RVLxA2X1, RVLxB2X1, -15.0D00, RVLxH4X1, RVLxD2)
          RVLxA2H1 = 2.0D00 * RVLxBB1 * RVCxFB(2) / RVCxHB(1)
          RVLxB2H1 = -RVLxA2H1 * RVCxHB(1)
          CALL DH4DP(RVLxA2H1, RVLxB2H1, 0.0D00, RVLxH4H1, RVLxD2)
          RVLxA2X2 = RVLxBB2
          RVLxB2X2 = -2.0D00 * RVCxHB(2) * RVLxBB2
          RVLxC2X2 = (-RVLxB2X2 * RVCxHB(2)/2.0D00) - 1.0D00
          CALL DH4DP(RVLxA2X2, RVLxB2X2, RVLxC2X2, RVLxH4X2, RVLxD2)
          RVLxB2H2 = -2.0D00 * RVCxFB(3) * RVLxBB2
          RVLxC2H2 = - RVLxB2H2 * RVCxHB(2)
          CALL DH4DP(0.0D00, RVLxB2H2, RVLxC2H2, RVLxH4H2, RVLxD2)
          RVLxA2Y2 = RVLxB2H2 / RVCxHB(3)
          RVLxB2Y2 = -2.0D00 * RVLxA2Y2 * RVCxHB(2)
          RVLxC2Y2 = RVLxA2Y2 * RVCxHB(2) * RVCxHB(2)
          CALL DH4DP(RVLxA2Y2, RVLxB2Y2, RVLxC2Y2, RVLxH4Y2, RVLxD2
          RVSxDXHB = RVLxH4X1*RVCxB5(2) + RVLxH4X2*RVCxB5(3) +
     #RVLxH4H1*RVCxB6(1)
          RVSxDXHB = RVSxDXHB + RVLxH4H2*RVCxB6(2) + RVLxH4Y2*RVCxB6(3
          RVSxDXHB = -RVSxDXHB / RVCxV1
          RVSxDYHB = RVLxH4X1*RVCxE5(2) + RVLxH4X2*RVCxE5(3) +
     #RVLxH4H1*RVCxE6(1)
          RVSxDYHB = RVSxDYHB + RVLxH4H2*RVCxE6(2) + RVLxH4Y2*RVCxE6(3
          RVSxDYHB = RVSxDYHB / RVCxV2
          RETURN
          END
```

H4TLT Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
IVSXSN.	param		

H4TLT Local Symbols

Name	Class	Type	Size
RVLXH4H1.	local	REAL*8	8
RVLXBB1	local	REAL*8	8
RVLXH4H2.	local	REAL*8	8
RVLXBB2	local	REAL*8	8
RVLXD2.	local	REAL*8	8
RVLXA2X1.	local	REAL*8	8
RVLXA2X2.	local	REAL*8	8
RVLXB2X1.	local	REAL*8	8
RVLXA2Y2.	local	REAL*8	8
RVLXB2X2.	local	REAL*8	8
RVLXB2Y2.	local	REAL*8	8
RVLXC2X2.	local	REAL*8	8
RVLXC2Y2.	local	REAL*8	8
RVLXH4X1.	local	REAL*8	8
RVLXH4X2.	local	REAL*8	8
RVLXH4Y2.	local	REAL*8	8
RVLXA2H1.	local	REAL*8	8
RVLXB2H1.	local	REAL*8	8
RVLXB2H2.	local	REAL*8	8
RVLXC2H2.	local	REAL*8	8
ITCXA	IONO3	INTEGER*4	4
RVCXA2.	VAR4	REAL*8	8
RVCXB2.	VAR4	REAL*8	8
RVCXC2.	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHB.	IONO1	REAL*8	24
RVCXFB.	IONO1	REAL*8	24
RVCXHT3	VAR4	REAL*8	8
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8

```
Line#  Source Line
      SUBROUTINE DH4DP(RVSxA2D, RVSxB2D, RVSxC2D, RVSxH4D, RVSxD2)
C
C-----------------------------------------------------------------
C
C     DH4DP  -- SUBROUTINE TO CALCULATE THE DERIVATIVE OF THE
C               BOUNDARY HEIGHT W.R.T. A PARTICULAR PARAMETER.
C
C
C     CALLED BY:     H4TLT
C
C-----------------------------------------------------------------
C
C     AUTHOR:        MICHAEL H. REILLY & ERIC L. STROBEL
C
C     DATE:          09/30/87
C
C     VERSION:       1.0
C
C-----------------------------------------------------------------
C
C     REVISED:       09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C     USES:          RVSxA,B,C2D   The partials of the para-
C                                  bolic profile parameters w.r.t.
C                                  the parameter of interest.
C
C                    RVSxD2        The discriminant of the para-
C                                  bolic parameters.
C
C
C     To calculate the partial derivative of H4 with respect to
C       the chosen parameter.  For further details, see the
C       RADAR-C report referred to in the documentation.
C
C
C     RETURNS:       RVSxH4D        The aforementioned partial
C                                  derivative.
C
C-----------------------------------------------------------------
C
      REAL*8 RVSxA2D, RVSxB2D, RVSxC2D, RVSxH4D, RVSxD2, RVCxHT4
      REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3
C
      COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4
C
      RVSxH4D = (-RVCxHT4 + RVCxC2/RVSxD2) * RVSxA2D
      RVSxH4D = RVSxH4D - (1.0D00 + RVCxB2/RVSxD2)*RVSxB2D/2.0D00
      RVSxH4D = (RVSxH4D + (RVCxA2/RVSxD2)*RVSxC2D)/RVCxA2
```

```
Line#  Source Line

  533          RETURN
  534          END
```

DH4DP Local Symbols

Name	Class	Type	Size
RVSXD2.	param		
RVSXH4D	param		
RVSXC2D	param		
RVSXB2D	param		
RVSXA2D	param		
RVCXC2.	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXA2.	VAR4	REAL*8	8
RVCXB2.	VAR4	REAL*8	8

Line# Source Line

```
      SUBROUTINE H1LTLT(RVSxDXHB, RVSxDYHB)
C
C---------------------------------------------------------------------
C
C     H1LTLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
C                   LINEAR VALLEY - LINEAR F1 BOUNDARY
C
C     CALLED BY:      TILTS
C
C     CALLS:          DH1LDP
C
C---------------------------------------------------------------------
C
C     AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C     DATE:           09/30/87
C
C     VERSION:        1.0
C
C---------------------------------------------------------------------
C
C     REVISED:        09/30/87 -- V1.0.  Initial revision.
C
C---------------------------------------------------------------------
C
C     USES:           The parameters from the COMMON blocks.
C
C
C     To calculate the rate of change of the boundary in the
C       locally horizontal directions.  The boundary here is
C       the boundary between the linear valley profile and the
C       linear F1 profile.  The routine DH1LDP is just
C       d (H1L) / d (parameter).  For further details, see the
C       RADAR-C report referred to in the documentation.
C
C
C     RETURNS:        RVSxDXHB         (d/dx) Height of Boundary.
C
C                     RVSxDYHB         (d/dy) Height of Boundary.
C
C---------------------------------------------------------------------
C
      INTEGER ITCxA
C
      REAL*8 RVSxDXHB, RVSxDYHB, RVLxBB1, RVCxB0, RVCxHU, RVCxHL
      REAL*8 RVLxHUXE, RVCxHB(3), RVCxFB(3),RVLxYSX1, RVCxH2
      REAL*8 RVLxBB2, RVCxYS, RVLxA0XE, RVCxA0, RVLxB0XE, RVLxH1XE
      REAL*8 RVLxS1X1, RVLxH1X1, RVLxS1H1, RVLxH1H1, RVLxHUX2
      REAL*8 RVLxA0X2, RVLxB0X2, RVLxS1X2, RVLxH1X2, RVLxA0H2
      REAL*8 RVLxB0H2, RVLxS1H2, RVLxH1H2, RVLxHUY2, RVLxA0Y2
```

Line# Source Line

```
      REAL*8 RVLxB0Y2, RVLxS1Y2, RVLxH1Y2, RVCxB5(3), RVCxB6(3)
      REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxALPH
      REAL*8 RVCxBETA, RVCxET1, RVCxET2, RVCxA5(3), RVCxA6(3)
      REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU, RVCxHB1
      REAL*8 RTCxA, RTCxB, RTCxC, RTCxD, RTCxE
C
      COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
      COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
      COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
      COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
      COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
      COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RTCxC,RTCxD,RTCxE
C
      RVLxBB1 = RVCxB0 / (RVCxHU - RVCxHL)
      RVLxHUXE = 0.9604D00 * RVCxHB(3) * RVCxHB(3) / (2.0D00 *
     #RVCxFB(3) * (RVCxHB(2) - RVCxHU))
      RVLxYSX1 = RVCxHB(3) * RVCxHB(3) / (2.0D00 * RVCxFB(3) *
     #(RVCxHB(2) - RVCxH2))
      RVLxBB2 = RVCxFB(2) / (RVCxYS * RVCxYS)
      RVLxA0XE = RVCxA0/RVCxFB(1) + RVCxHL*RVLxBB1*RVLxHUXE
      RVLxB0XE = RVCxB0/RVCxFB(1) - RVLxBB1*RVLxHUXE
      CALL DH1LDP(RVLxA0XE, 0.0D00, RVLxB0XE, 0.0D00, RVLxH1XE)
      RVLxS1X1 = 1.0D00 / RVCxYS
      IF (RVCxYS.NE.1.0D00) RVLxS1X1 = RVLxS1X1 - RVLxBB2*RVLxYSX1
      CALL DH1LDP(0.0D00,0.0D00,0.0D00, RVLxS1X1, RVLxH1X1)
      RVLxS1H1 = 0.0D00
      IF (RVCxYS.NE.1.0D00) RVLxS1H1 = 0.75D00 * RVLxBB2
      CALL DH1LDP(0.0D00,1.0D00,0.0D00, RVLxS1H1, RVLxH1H1)
      RVLxHUX2 = -RVLxHUXE * RVCxFB(1) / RVCxFB(3)
      RVLxA0X2 = RVCxHL * RVLxBB1 * RVLxHUX2
      RVLxB0X2 = -RVLxBB1 * RVLxHUX2
      RVLxS1X2 = 0.0D00
      IF (RVCxYS.NE.1.0D00) RVLxS1X2 = RVLxBB2*RVLxYSX1*RVCxFB(2)
     #/RVCxFB(3)
      CALL DH1LDP(RVLxA0X2, 0.0D00, RVLxB0X2, RVLxS1X2, RVLxH1X2)
      RVLxA0H2 = RVCxHL * RVLxBB1
      RVLxB0H2 = -RVLxBB1
      RVLxS1H2 = 0.0D00
      IF (RVCxYS.NE.1.0D00) RVLxS1H2 = -RVLxBB2
      CALL DH1LDP(RVLxA0H2, 0.0D00, RVLxB0H2, RVLxS1H2, RVLxH1H2)
      RVLxHUY2 = -(RVCxHB(2) - RVCxHU) / RVCxHB(3)
      RVLxA0Y2 = RVCxHL * RVLxBB1 * RVLxHUY2
      RVLxB0Y2 = -RVLxBB1 * RVLxHUY2
      RVLxS1Y2 = 0.0D00
      IF (RVCxYS.NE.1.0D00) RVLxS1Y2 = RVLxBB2*(RVCxHB(2)-RVCxH2)
     #/RVCxHB(3)
      CALL DH1LDP(RVLxA0Y2, 0.0D00, RVLxB0Y2, RVLxS1Y2, RVLxH1Y2)
      RVSxDXHB = RVLxH1XE*RVCxB5(1) + RVLxH1X1*RVCxB5(2)
     #+ RVLxH1X2*RVCxB5(3)
      RVSxDXHB = RVSxDXHB + RVLxH1H1*RVCxB6(1) + RVLxH1H2*RVCxB6(2)
```

```
Line#  Source Line

          #+ RVLxH1Y2*RVCxB6(3)
             RVSxDXHB = -RVSxDXHB / RVCxV1
             RVSxDYHB = RVLxH1XE*RVCxE5(1) + RVLxH1X1*RVCxE5(2)
          #+ RVLxH1X2*RVCxE5(3)
             RVSxDYHB = RVSxDYHB + RVLxH1H1*RVCxE6(1) + RVLxH1H2*RVCxE6(2)
          #+ RVLxH1Y2*RVCxE6(3)
             RVSxDYHB = RVSxDYHB / RVCxV2
             RETURN
             END
```

H1LTLT Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
RVLXYSX1.	local	REAL*8	8
RVLXBB1	local	REAL*8	8
RVLXBB2	local	REAL*8	8
RVLXHUXE.	local	REAL*8	8
RVLXA0X2.	local	REAL*8	8
RVLXA0Y2.	local	REAL*8	8
RVLXB0X2.	local	REAL*8	8
RVLXB0Y2.	local	REAL*8	8
RVLXS1H1.	local	REAL*8	8
RVLXS1H2.	local	REAL*8	8
RVLXH1X1.	local	REAL*8	8
RVLXH1X2.	local	REAL*8	8
RVLXH1Y2.	local	REAL*8	8
RVLXS1X1.	local	REAL*8	8
RVLXS1X2.	local	REAL*8	8
RVLXA0XE.	local	REAL*8	8
RVLXS1Y2.	local	REAL*8	8
RVLXB0XE.	local	REAL*8	8
RVLXH1XE.	local	REAL*8	8
RVLXHUX2.	local	REAL*8	8
RVLXHUY2.	local	REAL*8	8
RVLXA0H2.	local	REAL*8	8
RVLXB0H2.	local	REAL*8	8
RVLXH1H1.	local	REAL*8	8
RVLXH1H2.	local	REAL*8	8
RTCXE	VAR2	REAL*8	8
ITCXA	IONO3	INTEGER*4	4
RVCXB0.	VAR1	REAL*8	8
RVCXHU.	VAR1	REAL*8	8
RVCXHL.	TEMP1	REAL*8	8
RVCXHB.	IONO1	REAL*8	24
RVCXFB.	IONO1	REAL*8	24
RVCXH2.	VAR2	REAL*8	8

H1LTLT Local Symbols

Name	Class	Type	Size
RVCXYS.	VAR2	REAL*8	8
RVCXA0.	VAR1	REAL*8	8
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXK1.	TEMP1	REAL*8	8
RVCXXL.	VAR1	REAL*8	8
RVCXXU.	VAR1	REAL*8	8
RVCXHB1	VAR2	REAL*8	8
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8
RTCXC	VAR2	REAL*8	8
RTCXD	VAR2	REAL*8	8

Line# Source Line

```
      SUBROUTINE DH1LDP(RVSxA0P, RVSxH1D, RVSxB0P, RVSxS1P, RVSxDD)
C
C------------------------------------------------------------------------
C
C        DH1LDP  -- SUBROUTINE TO CALCULATE THE DERIVATIVE OF THE
C                   BOUNDARY HEIGHT W.R.T. A PARTICULAR PARAMETER.
C
C
C        CALLED BY:       H1LTLT
C
C------------------------------------------------------------------------
C
C        AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
C
C        DATE:            09/30/87
C
C        VERSION:         1.0
C
C------------------------------------------------------------------------
C
C        REVISED:         09/30/87 -- V1.0.  Initial revision.
C
C------------------------------------------------------------------------
C
C        USES:            RVSxA,B0P      The partials of the linear
C                                       valley profile parameters w.r.t.
C                                       the parameter of interest.
C
C                         RVSxH1D,S1P    The partials of the linear
C                                       F1 profile parameters w.r.t.
C                                       the parameter of interest.
C
C
C        To calculate the partial derivative of H1L with respect to
C          the chosen parameter.  For further details, see the
C          RADAR-C report referred to in the documentation.
C
C
C        RETURNS:         RVSxDD         The aforementioned partial
C                                       derivative.
C
C------------------------------------------------------------------------
C
      REAL*8 RVSxA0P, RVSxH1D, RVSxB0P, RVSxS1P, RVSxDD, RTLxL
      REAL*8 RVCxSL1, RVCxB0, RVCxHB1, RVCxA0, RVCxXL, RVCxXU
      REAL*8 RVCxHU, RVCxH2, RVCxYS, RTCxA, RTCxB
C
      COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
      COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RVCxSL1,RTCxA,RTCxB
C
```

Line#   Source Line

```
           RTLx1 = RVCxSL1 - RVCxB0
           RVSxDD = RTLx1 * (RVSxA0P + 0.75D00*RVCxSL1*RVSxH1D)
           RVSxDD = RVSxDD + (RVCxHB1*RVCxSL1 + RVCxA0) * RVSxB0P
           RVSxDD = (RVSxDD - (RVCxHB1*RVCxB0 + RVCxA0)*RVSxS1P)
     */ (RTLx1 * RTLx1)
           RETURN
           END
```

DH1LDP   Local Symbols

Name	Class	Type	Size
RVSXDD	param		
RVSXS1P	param		
RVSXB0P	param		
RVSXH1D	param		
RVSXA0P	param		
RTLX1	local	REAL*8	8
RVCXSL1	VAR2	REAL*8	8
RVCXB0	VAR1	REAL*8	8
RVCXHB1	VAR2	REAL*8	8
RVCXA0	VAR1	REAL*8	8
RVCXXL	VAR1	REAL*8	8
RVCXXU	VAR1	REAL*8	8
RVCXHU	VAR1	REAL*8	8
RVCXH2	VAR2	REAL*8	8
RVCXYS	VAR2	REAL*8	8
RTCXA	VAR2	REAL*8	8
RTCXB	VAR2	REAL*8	8

Line# Source Line

```
      SUBROUTINE H2TLT(RVSxDXHB, RVSxDYHB)
C
C-----------------------------------------------------------------
C
C     H2TLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
C                     LINEAR F1 - F2 BOUNDARY
C
C     CALLED BY:      TILTS
C
C-----------------------------------------------------------------
C
C     AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C     DATE:           09/30/87
C
C     VERSION:        1.0
C
C-----------------------------------------------------------------
C
C     REVISED:        09/30/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C     USES:           The parameters from the COMMON blocks.
C
C
C     To calculate the rate of change of the boundary in the
C       locally horizontal directions.  The boundary here is
C       the boundary between the linear F1 profile and the
C       parabolic F2 profile.  For further details, see the
C       RADAR-C report referred to in the documentation.
C
C
C     RETURNS:        RVSxDXHB        (d/dx) Height of Boundary.
C
C                     RVSxDYHB        (d/dy) Height of Boundary.
C
C-----------------------------------------------------------------
C
      INTEGER ITCxA
C
      REAL*8 RVSxDXHB, RVSxDYHB, RVLxH2X1, RVCxHB(3), RVCxFB(3)
      REAL*8 RVCxH2, RVLxH2X2, RVLxH2H2, RVLxH2Y2, RVCxB5(3)
      REAL*8 RVCxB6(3), RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2
      REAL*8 RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVCxA5(3)
      REAL*8 RVCxA6(3), RVCxHB1, RTCxA, RTCxB, RTCxC, RTCxD
      REAL*8 RTCxE, RTCxF
C
      COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
      COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
```

Line# Source Line

```
      COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
      COMMON /VAR2/ RVCxHB1,RVCxH2,RTCxC,RTCxD,RTCxE,RTCxF
C
      RVLxH2X1 = (RVCxHB(3) * RVCxHB(3))/(2.0D00 * RVCxFB(3) *
     #(RVCxHB(2) - RVCxH2))
      RVLxH2X2 = -RVLxH2X1 * RVCxFB(2) / RVCxFB(3)
      RVLxH2H2 = 1.0D00
      RVLxH2Y2 = -(RVCxHB(2) - RVCxH2) / RVCxHB(3)
      RVSxDXHB = RVLxH2X1*RVCxB5(2) + RVLxH2X2 * RVCxB5(3)
      RVSxDXHB = RVSxDXHB + RVLxH2H2*RVCxB6(2) + RVLxH2Y2*RVCxB6(3)
      RVSxDXHB = -RVSxDXHB / RVCxV1
      RVSxDYHB = RVLxH2X1*RVCxE5(2) + RVLxH2X2 * RVCxE5(3)
      RVSxDYHB = RVSxDYHB + RVLxH2H2*RVCxE6(2) + RVLxH2Y2*RVCxE6(3)
      RVSxDYHB = RVSxDYHB / RVCxV2
      RETURN
      END
```

H2TLT Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
RVLXH2H2.	local	REAL*8	8
RVLXH2X1.	local	REAL*8	8
RVLXH2X2.	local	REAL*8	8
RVLXH2Y2.	local	REAL*8	8
ITCXA	IONO3	INTEGER*4	4
RVCXHB.	IONO1	REAL*8	24
RVCXFB.	IONO1	REAL*8	24
RVCXH2.	VAR2	REAL*8	8
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXHB1	VAR2	REAL*8	8
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8
RTCXC	VAR2	REAL*8	8
RTCXD	VAR2	REAL*8	8
RTCXE	VAR2	REAL*8	8

H2TLT Local Symbols

Name	Class	Type	Size
RTCXF	VAR2	REAL*8	8

Global Symbols

Name	Class	Type	Size
DH1LDP.	FSUBRT	***	***
DH1PDP.	FSUBRT	***	***
DH4DP	FSUBRT	***	***
H1LTLT.	FSUBRT	***	***
H1PTLT.	FSUBRT	***	***
H2TLT	FSUBRT	***	***
H4TLT	FSUBRT	***	***
H5TLT	FSUBRT	***	***
HUTLT	FSUBRT	***	***
INTSIGN	extern	INTEGER*4	***
IONO1	common	***	80
IONO2	common	***	144
IONO3	common	***	36
PRAM.	common	***	32
RAID.	common	***	4
TEMP1	common	***	32
TILTS	FSUBRT	***	***
VAR1.	common	***	40
VAR2.	common	***	48
VAR3.	common	***	40
VAR4.	common	***	40

Code size = 13f3 (5107)
Data size = 007a (122)

Line# Source Line

```
      SUBROUTINE NEWCS(IFSxG)
C
C-----------------------------------------------------------------
C
C   NEWCS -- SUBROUTINE TO CALCULATE NEW CX, CY, CZ VALUES
C
C       CALLED BY:      RAYSUB
C
C       CALLS:          FREESP
C
C-----------------------------------------------------------------
C
C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
C
C       DATE:           09/17/86
C
C       VERSION:        2.0
C
C-----------------------------------------------------------------
C
C       REVISED:        07/25/86 -- INITIAL REVISION.  TRANSLATED
C                       FROM TEKTRONIX BASIC TO VAX FORTRAN BY
C                       ERIC L. STROBEL.
C
C                       07/30/86 -- V1.1.  Change over to use of
C                       REAL*8 precision in the calculations.
C
C                       09/17/86 -- V2.0.  Got rid of the use of
C                       approximate rotation matrix in calc. of
C                       the new c-values.
C
C-----------------------------------------------------------------
C
C      USES:   IFSxG            A FLAG
C              IVCxSX,SY,SZ     OLD SIGN VARIABLES
C              RVCxXF,YF,ZF     FINAL INTERVAL COORDINATES
C              RVCxLA1          PREVIOUS POINT'S LAT
C              RVCxLO1          PREVIOUS POINT'S LON
C              RVCxLAI          LAT OF BEGINNING OF INTERVAL
C              RVCxLOI          LON OF BEGINNING OF INTERVAL
C              RVCxALPH         X
C              RVCxBETA         X-IONOSPHERIC PARAMETERS
C              RVCxET1          X
C              RVCxET2          X
C              RVCxCX,CY,CZ     OLD C-VALUES
C
C       TO CALCULATE THE NEW C AND S VALUES.
C
C       RETURNS:
C              IVCxSX,SY,SZ     THE NEW SIGN VALUES
```

Line#   Source Line

```
C                 RVCxCX,CY,CZ    THE NEW C VALUES
C
C-----------------------------------------------------------------
C
      INTEGER IFSxG, IVCxSX, IVCxSY, IVCxSZ
C
      REAL*8 RVCxCX, RVCxCY, RVCxCZ, RVCxXF, RVCxYF, RVCxZF
      REAL*8 RVCxH5, RVCxLA1, RVCxLO1, RVCxHBOT
      REAL*8 RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVLxI(5), RVLxJ(5),
      REAL*8 RVLxK(5), RTLxC5, RTLxC6, RTLxC7, RTLxA2, RTLxA3
      REAL*8 RTLxA4, RTLxQ5, RTLxQ6, RTLxQ7, RTLxD6, RVCxD6
      REAL*8 RTCxA, RTCxB, RTCxC, RTCxD
C
      COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ, RVCxCX, RVCxCY, RVCxCZ
      COMMON /OTHER/ RVCxLA1,RVCxLO1,RVCxHBOT,RTCxA,RTCxB,RTCxC
      COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RVCxH5
      COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2
      COMMON /GORP/ RVCxD6, RTCxD
C
10000 RTLxC5 = RVCxCX
      RTLxC6 = RVCxCY
      RTLxC7 = RVCxCZ
      RVLxI(4) = DSIN(RVCxLA1)
      RVLxI(5) = DCOS(RVCxLA1)
C
C     A routine to concoct the rotation matrix for the
C       calculation of the new C values.
C
      CALL FREESP(RVLxI, RVLxJ, RVLxK)
C
C     Recall that the flag IFSxG can take on values of 3, 4, & 5,
C        corresponding to the occurrance of reflections in the
C        z, x, & y directions, respectively.
C
      IF (IFSxG.EQ.4) THEN
          RTLxA4 = 0.0D00
      ELSE
          RTLxA4 = IVCxSX*DSQRT(RTLxC5 - RVCxET1*RVCxXF)
      ENDIF
      IF (IFSxG.EQ.5) THEN
          RTLxA2 = 0.0D00
      ELSE
          RTLxA2 = IVCxSY*DSQRT(RTLxC6 - RVCxET2*RVCxYF)
      ENDIF
      IF (IFSxG.EQ.3) THEN
          RTLxA3 = 0.0D00
      ELSE
          RTLxA3 = IVCxSZ*DSQRT(RVCxBETA*RVCxZF*RVCxZF -
     *RVCxALPH*RVCxZF + RTLxC7)
      ENDIF
```

```
Line#  Source Line

C
              RTLxQ5 = RVLxI(1)*RTLxA4 + RVLxI(2)*RTLxA2 + RVLxI(3)*RTLxA3
              RTLxQ6 = RVLxJ(1)*RTLxA4 + RVLxJ(2)*RTLxA2 + RVLxJ(3)*RTLxA3
              RTLxQ7 = RVLxK(1)*RTLxA4 + RVLxK(2)*RTLxA2 + RVLxK(3)*RTLxA3
              RTLxD6 = DSQRT(RVCxXF*RVCxXF + RVCxYF*RVCxYF + RVCxZF*RVCxZF)
              RVCxD6 = RTLxD6
              IF (IFSxG.NE.4) GO TO 10414
C
C             Begin handling for an x-reflection.
C
              IF (INTSIGN(RTLxQ5).EQ.IVCxSX) GO TO 10410
              IVCxSX = -IVCxSX
              GO TO 10416
C
C             A limit is set to ensure that the calculation doesn't take
C               an infinite time to creep up to a reflection point, so
C               when the limit is reached, a reflection is forced.
C
10410         IF (RTLxD6.GT.1.0D-04) GO TO 10416
              IVCxSX = -IVCxSX
              RTLxQ5 = 0.0D00
              GO TO 10416
10414         IVCxSX = INTSIGN(RTLxQ5)
              GO TO 10419
C
C             The x-reflection is complete, so handle updating the values
C               in the y and z directions.
C
10416         IVCxSY = INTSIGN(RTLxQ6)
              IVCxSZ = INTSIGN(RTLxQ7)
              GO TO 10446
10419         IF (IFSxG.NE.5) GO TO 10428
C
C             Begin handling for a y-reflection.
C
              IF (INTSIGN(RTLxQ6).EQ.IVCxSY) GO TO 10424
              IVCxSY = -IVCxSY
              GO TO 10430
C
C             Again a limit is set to ensure that the calculation doesn't
C               take an infinite time to creep up to a reflection point,
C               so when the limit is reached, a reflection is forced.
C
10424         IF (RTLxD6.GT.1.0D-04) GO TO 10430
              IVCxSY = -IVCxSY
              RTLxQ6 = 0.0D00
              GO TO 10430
10428         IVCxSY = INTSIGN(RTLxQ6)
              GO TO 10433
C
```

```
Line#  Source Line

       C        The y-reflection is complete, so handle updating the values
       C          in the x and z directions.
       C
       10430    IVCxSX = INTSIGN(RTLxQ5)
                IVCxSZ = INTSIGN(RTLxQ7)
                GO TO 10446
       10433    IF (IFSxG.NE.3) GO TO 10442
       C
       C        Begin handling for a z-reflection.
       C
                IF (INTSIGN(RTLxQ7).EQ.IVCxSZ) GO TO 10438
       10436    IVCxSZ = -IVCxSZ
                GO TO 10444
       C
       C        Again a limit is set to ensure that the calculation doesn't
       C          take an infinite time to creep up to a reflection point,
       C          so when the limit is reached, a reflection is forced.
       C
       10438    IF (RTLxD6.GT.1.0D-04) GO TO 10444
                IVCxSZ = -IVCxSZ
                RTLxQ7 = 0.0D00
                GO TO 10444
       10442    IVCxSZ = INTSIGN(RTLxQ7)
                GO TO 10446
       C
       C        The z-reflection is complete, so hanile updating the values
       C          in the x and y directions.
       C
       10444    IVCxSX = INTSIGN(RTLxQ5)
                IVCxSY = INTSIGN(RTLxQ6)
       10446    RVCxCX = RTLxQ5 * RTLxQ5
                RVCxCY = RTLxQ6 * RTLxQ6
                RVCxCZ = RTLxQ7 * RTLxQ7
       C
       C        IFSxG = 6 which means that the ray is outside the bounds
       C          of the ionosphere, headed down, so presumably the program
       C          is handling an earth bounce event.  Therefore, a specular
       C          reflection is forced.
       C
                IF (IFSxG.EQ.6.AND.RVCxH5.LT.RVCxHBOT) IVCxSZ = -IVCxSZ
                RETURN
                END
```

EWCS Local Symbols

ame	Class	Type	Size
FSXG	param		
TLXC5.	local	REAL*8	8

NEWCS Local Symbols

Name	Class	Type	Size
RTLXC6	local	REAL*8	8
RTLXD6	local	REAL*8	8
RTLXC7	local	REAL*8	8
RTLXQ5	local	REAL*8	8
RTLXQ6	local	REAL*8	8
RTLXQ7	local	REAL*8	8
RVLXI	local	REAL*8	40
RVLXJ	local	REAL*8	40
RVLXK	local	REAL*8	40
RTLXA2	local	REAL*8	8
RTLXA3	local	REAL*8	8
RTLXA4	local	REAL*8	8
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
IVCXSZ	MORE	INTEGER*4	4
RVCXCX	MORE	REAL*8	8
RVCXCY	MORE	REAL*8	8
RVCXCZ	MORE	REAL*8	8
RVCXXF	END	REAL*8	8
RVCXYF	END	REAL*8	8
RVCXZF	END	REAL*8	8
RVCXH5	END	REAL*8	8
RVCXLA1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXD6	GORP	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	OTHER	REAL*8	8
RTCXD	GORP	REAL*8	8

```
Line#  Source Line

          SUBROUTINE TRIANG
      C
      C-------------------------------------------------------------------
      C
      C   TRIANG -- SUBROUTINE TO CHECK IF A LOCATION IS WITHIN A
      C                   TRIANGLE OF S.C.P.'S
      C
      C         CALLED BY:        IONOPAR
      C
      C-------------------------------------------------------------------
      C
      C         AUTHOR:           MICHAEL H. REILLY & ERIC L. STROBEL
      C
      C         DATE:             02/08/88
      C
      C         VERSION:          2.2
      C
      C-------------------------------------------------------------------
      C
      C         REVISED:          07/25/86 -- INITIAL REVISION.  TRANSLATED
      C                           FROM TEKTRONIX BASIC TO VAX FORTRAN BY
      C                           ERIC L. STROBEL.
      C
      C                           07/30/86 -- V1.1.  Change over to use of
      C                           REAL*8 precision in the calculations.
      C
      C                           10/10/86 -- V2.0.  Changed to conform to
      C                           the change over to RAYTRACE as a subrout.
      C
      C                           09/01/87 -- V2.1.  Slight changes to speed
      C                           things up a bit.
      C
      C                           02/08/88 -- V2.2.  More streamlining of INTs
      C                           and NINTs.  Logical variable added to help
      C                           correctly handle the (possibly) worrisome
      C                           case of the grid at the geographic north pole.
      C
      C-------------------------------------------------------------------
      C
      C        USES:   IVSxSCT1,2,3     THE THREE S.C.P.'S THAT MAKE THE
      C                                             TRIANGLE
      C                RVCxX,Y,Z         INTERMEDIATE COORDINATES
      C                RVCxU,V,W(900)   THE S.C.P. COORDINATES
      C
      C
      C         TO DETERMINE WHETHER OR NOT A LOCATION IS WITHIN A
      C             TRIANGLE OF S.C.P.'S.  First, the locations  & array
      C             indices) of the four surrounding grid points are
      C             obtained, and then the potential triangles (involving
      C             three of the corners of this surrounding rectangle
```

Line# Source Line

```
C             are checked in an attempt to make sure that the point
C             of interest is as far within a triangle as possible.
C             The three grid points that make up this triangle are
C             reported.  These are the points from which the
C             interpolation is based.  Recall the following
C             definitions: RVxGRID(1) = lat spacing
C                                 (2) = lon spacing
C                                 (3) = starting lat
C                                 (4) = starting lon
C                                 (5) = # in lat
C                                 (6) = # in lon.
C
C
C         RETURNS:
C                IFSxN4          FLAG (=1 IF NOT INTERIOR TO A
C                                         TRIANGLE OF S.C.P.'S)
C
C-----------------------------------------------------------------
C
        INTEGER IVLxT1, IVLxT2, IVLxT3, IVLxT4, IVCxSCT1
        INTEGER IVCxSCT2, IVCxSCT3, IVCxSCS, IVLxNLN, IVLxILN
C
        REAL*8 RVLxLAIN, RVLxLOIN, RVCxLAI, RVxGRID(6), RVCxLOI
        REAL*8 RVLxP1, RVLxP2, RVCxXI, RVCxYI, RVCxZI
C
        LOGICAL LVLxPOLE
C
        COMMON /MAINDAT/ RVxGRID
        COMMON /START/ RVCxXI, RVCxYI, RVCxZI, RVCxLAI, RVCxLOI
        COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
C
        RVLxLAIN = (RVCxLAI - RVxGRID(3)) / RVxGRID(1)
        RVLxLOIN = (RVCxLOI - RVxGRID(4)) / RVxGRID(2)
        IVLxILN = INT(RVLxLAIN)
        RVLxP1 = RVLxLAIN - IVLxILN
        RVLxP2 = RVLxLOIN - INT(RVLxLOIN)
        IVLxNLN = NINT(RVxGRID(6))
        IVLxT1 = IVLxILN * IVLxNLN + INT(RVLxLOIN) - 1
        IVLxT2 = IVLxT1 + 1
        IVLxT3 = IVLxT1 + IVLxNLN
        IVLxT4 = IVLxT3 + 1
        LVLxPOLE = NINT(RVxGRID(3) + (IVLxILN + 1) * RVxGRID(1)).EQ.90
        IF (RVLxP1.LT.0.5.OR.LVLxPOLE) THEN
            IVCxSCT1 = IVLxT1
            IVCxSCT2 = IVLxT2
            IF (RVLxP2.LT.0.5D00) THEN
                IVCxSCT3 = IVLxT3
            ELSE
                IVCxSCT3 = IVLxT4
            ENDIF
```

```
Line#  Source Line

             ELSE
                 IVCxSCT2 = IVLxT3
                 IVCxSCT3 = IVLxT4
                 IF (RVLxP2.LT.0.5D00) THEN
                     IVCxSCT1 = IVLxT1
                 ELSE
                     IVCxSCT1 = IVLxT2
                 ENDIF
             ENDIF
             RETURN
             END
```

TRIANG Local Symbols

Name	Class	Type	Size
IVLXT1.	local	INTEGER*4	4
IVLXT2.	local	INTEGER*4	4
IVLXT3.	local	INTEGER*4	4
IVLXT4.	local	INTEGER*4	4
RVLXP1.	local	REAL*8	8
RVLXP2.	local	REAL*8	8
IVLXILN	local	INTEGER*4	4
IVLXNLN	local	INTEGER*4	4
RVLXLAIN.	local	REAL*8	8
LVLXPOLE.	local	LOGICAL*4	4
RVLXLOIN.	local	REAL*8	8
IVCXSCT1.	TEMP2	INTEGER*4	4
IVCXSCT2.	TEMP2	INTEGER*4	4
IVCXSCT3.	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
RVCXLAI	START	REAL*8	8
RVXGRID	MAINDAT	REAL*8	48
RVCXLOI	START	REAL*8	8
RVCXXI.	START	REAL*8	8
RVCXYI.	START	REAL*8	8
RVCXZI.	START	REAL*8	8

```
Line#  Source Line

      SUBROUTINE ACCFSP(RVSxH0, RVSxS4, IFSxG)
C
C-----------------------------------------------------------------------
C
C   ACCFSP -- SUBROUTINE TO SPEED UP CALCULATION OF FREE SPACE
C                PROPAGATION
C
C        CALLED BY: ENDPT
C
C-----------------------------------------------------------------------
C
C        AUTHOR:             MICHAEL H. REILLY & ERIC L. STROBEL
C
C        DATE:               03/18/88
C
C        VERSION:            2.2
C
C-----------------------------------------------------------------------
C
C        REVISED:            07/25/86 -- INITIAL REVISION.  TRANSLATED
C                            FROM TEKTRONIX BASIC TO VAX FORTRAN BY
C                            ERIC L. STROBEL.
C
C                            07/30/86 -- V1.1.  Change over to use of
C                            REAL*8 precision in the calculations.
C
C                            09/01/87 -- V2.0.  Changed to allow ray to
C                            be above the top of the ionosphere.
C
C                            12/08/87 -- V2.1.  A change has been made
C                            to make sure that the ray stops at the
C                            desired point.
C
C                            03/18/88 -- V2.2.  Minor changes made to
C                            accommodate the new usage of angular range.
C
C-----------------------------------------------------------------------
C
C       USES:   IVCxSX,SY,SZ     SIGN VARIABLES
C               RPCxRE           EARTH RADIUS
C               RVSxH0           CURRENT HEIGHT
C               RVCxCX,CY,CZ     C-VALUES
C
C        TO OBTAIN THE ENDPOINT OF A RAYPATH INCREMENT THAT IS
C               THROUGH FREESPACE.
C
C        RETURNS:
C               RVCxXF,YF,ZF     THE ENDPOINT COORDINATES
C
C-----------------------------------------------------------------------
```

Line# Source Line

```
C
          INTEGER IVCxSX, IVCxSY, IVCxSZ, IVLxSSS, IVLxJ, IVLxI
          INTEGER IFSxG
C
          REAL*8 RPCxRE, RVSxH0, RVCxCX, RVCxCY, RVCxCZ, RVLxHGT
          REAL*8 RVCxXF, RVCxYF, RVCxZF, RPCxPI, RTLxD, RVCxHCT
          REAL*8 RPCxDTR, RVCxH5, RTLxQ1, RTLxQ2, RTLxQ3, RPCxHTP
          REAL*8 RTLxT1, RTLxC, RTLxC2, RTLxRR(4), RVLxS,VLxRF
          REAL*8 RTLxL1,RTLxL2, RTLxL4,RTLxL5, RVCxHB,RTLxD1,RTLxD2
          REAL*8 RTLxD3, RVSxS4, RTLxZZZ, RVCxALIM, RTLxT2, RTLxT3
          REAL*8 RTLxRP, RTLxTTH, RTLxTHE, RTLxDEL, RTLxDP, RTLxS4
          REAL*8 RTLxSPR
C
          COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ, RVCxCX, RVCxCY, RVCxCZ
          COMMON /PRAM/ RPCxPI, RPCxDTR, RPCxRE, RPCxHTP
          COMMON /OTHER/ RTLxL1,RTLxL2,RVCxHB,RTLxL4,RTLxL5,RVCxHCT
          COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RVCxH5
          COMMON /GORP/ RTLxZZZ, RVCxALIM
C
          IVLxI = 0
C
C         Calculate the ray direction cosines.
C
          RTLxQ1 = IVCxSX*DSQRT(RVCxCX)
          RTLxQ2 = IVCxSY*DSQRT(RVCxCY)
          RTLxQ3 = IVCxSZ*DSQRT(RVCxCZ)
C
C         The calculation is the solution of the quadratic equation
C            resulting from a straight line piercing a sphere.  First,
C            the radii of the pertinent spheres are obtained, based on
C            whether the freespace propagation is above or below the
C            bounds of the model ionosphere.  Based on the discriminant
C            the proper solution is chosen.  The linear distance covered
C            is calculated (RVLxS) and using the direction cosines, the
C            final coordinate values are obtained.
C
          RTLxT1 = RVSxH0 + RPCxRE
          RTLxC =  RTLxT1*RTLxQ3
          RTLxC2 = RTLxC*RTLxC
          IVLxSSS = 1
          IF (RVCxH5.GE.RPCxHTP) THEN
              IVLxJ = 3
              RTLxRR(3) = RPCxHTP
              RTLxRR(4) = RVCxHCT
          ELSE
              IVLxJ = 1
              RTLxRR(1) = 0.0D00
              RTLxRR(2) = RVCxHB
          ENDIF
          IF (IVCxSZ.LT.0) GO TO 10000
```

```
Line#  Source Line

 9000    IVLxJ = IVLxJ + 1
         IVLxSSS = -1
10000    RTLxD = RTLxC2 + (RTLxRR(IVLxJ) - RVSxH0)*(RTLxT1 +
      #RTLxRR(IVLxJ) + RPCxRE)
         IF (RTLxD.LT.0.0D00) THEN
             PRINT *,' Ray passes through a minimum of altitude.'
             IFSxG = 7
             GO TO 9000
         ENDIF
         RVLxS = -RTLxC  - IVLxSSS * DSQRT(RTLxD)
11000    RVCxXF = RTLxQ1*RVLxS
         RVCxYF = RTLxQ2*RVLxS
         RVCxZF = RTLxQ3*RVLxS
C
C        DONE 12/8.  After some preliminary calculations, a check
C          is made to see if the straight line endpoint is beyond
C          the range cutoff.  If so, the distance traveled along
C          the line is backed off by an appropriate amount and the
C          endpoint coordinates are recalculated.
C
         RTLxT2 = RVCxZF + RTLxT1
         RTLxT3 = RVCxXF * RVCxXF + RVCxYF * RVCxYF
         RTLxRP = DSQRT(RTLxT3 + RTLxT2*RTLxT2)
         RTLxTTH = RVCxZF / DSQRT(RTLxT3)
         RTLxTTH = DATAN(RTLxTTH)
         RTLxTHE = RPCxPI / 2.0 + RTLxTTH
         RTLxDEL = RVLxS * DSIN(RTLxTHE) / RTLxRP
         RTLxDEL = DASIN(RTLxDEL)
         IF ((RVSxS4 + RTLxDEL).GE.RVCxALIM) THEN
             RTLxDP = RVCxALIM - RVSxS4
             RTLxSPR = RVLxS * RTLxT1 * DSIN(RTLxDP)
             RTLxSPR = RTLxSPR / (RTLxRP * DSIN(RTLxDEL - RTLxDP)
      # + RTLxT1 * DSIN(RTLxDP))
             RVCxXF = RTLxQ1*RTLxSPR
             RVCxYF = RTLxQ2*RTLxSPR
             RVCxZF = RTLxQ3*RTLxSPR
             RTLxT2 = RVCxZF + RTLxT1
             RTLxT3 = RVCxXF * RVCxXF + RVCxYF * RVCxYF
             RTLxRP = DSQRT(RTLxT3 + RTLxT2*RTLxT2)
             IF (DABS(RTLxRP-RPCxRE).GT.1.0D-02) THEN
                 IFSxG = 7
             ENDIF
         ENDIF
C
         RETURN
         END
```

ACCFSP Local Symbols

Name	Class	Type	Size
IFSXG	param		
RVSXS4	param		
RVSXH0	param		
RTLXQ1	local	REAL*8	8
IVLXI	local	INTEGER*4	4
RTLXC	local	REAL*8	8
RTLXQ2	local	REAL*8	8
IVLXJ	local	INTEGER*4	4
RTLXQ3	local	REAL*8	8
RTLXD	local	REAL*8	8
RTLXT1	local	REAL*8	8
RTLXT2	local	REAL*8	8
RTLXT3	local	REAL*8	8
RTLXDP	local	REAL*8	8
RTLXDEL	local	REAL*8	8
RVLXS	local	REAL*8	8
RTLXTHE	local	REAL*8	8
RTLXRP	local	REAL*8	8
RTLXRR	local	REAL*8	32
RTLXTTH	local	REAL*8	8
IVLXSSS	local	INTEGER*4	4
RTLXSPR	local	REAL*8	8
RTLXC2	local	REAL*8	8
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
IVCXSZ	MORE	INTEGER*4	4
RPCXRE	PRAM	REAL*8	8
RVCXCX	MORE	REAL*8	8
VCXCY	MORE	REAL*8	8
VCXCZ	MORE	REAL*8	8
VCXXF	END	REAL*8	8
VCXYF	END	REAL*8	8
VCXZF	END	REAL*8	8
PCXPI	PRAM	REAL*8	8
VCXHCT	OTHER	REAL*8	8
PCXDTR	PRAM	REAL*8	8
VCXH5	END	REAL*8	8
PCXHTP	PRAM	REAL*8	8
TLXL1	OTHER	REAL*8	8
TLXL2	OTHER	REAL*8	8
TLXL4	OTHER	REAL*8	8
TLXL5	OTHER	REAL*8	8
VCXHB	OTHER	REAL*8	8
TLXZZZ	GORP	REAL*8	8
VCXALIM	GORP	REAL*8	8

Line# Source Line

```
      SUBROUTINE FREESP(RVSxI, RVSxJ, RVSxK)
C
C-----------------------------------------------------------------
C
C   FREESP -- SUBROUTINE TO OBTAIN SOME VALUES USED TO COMPUTE
C                NEW C-VALUES IN THE CASE OF FREESPACE
C                PROPAGATION.
C
C        CALLED BY: NEWCS
C
C-----------------------------------------------------------------
C
C        AUTHOR:         MICHAEL H. REILLY
C
C        DATE:           07/30/86
C
C        VERSION:        1.1
C
C-----------------------------------------------------------------
C
C        REVISED:        07/25/86 -- INITIAL REVISION.  TRANSLATED
C                        FROM TEKTRONIX BASIC TO VAX FORTRAN BY
C                        ERIC L. STROBEL.
C
C                        07/30/86 -- V1.1.  Change over to use of
C                        REAL*8 precision in the calculations.
C
C-----------------------------------------------------------------
C
C       USES:    RVCxLO1        PREVIOUS POINT'S LON
C                RVCxLAI        LAT OF BEGINNING OF INTERVAL
C                RVCxLOI        LON OF BEGINNING OF INTERVAL
C                RVSxI(5)       X
C                RVSxJ(5)       X-ARRAYS OF INTERMEDIATE VALUES
C                RVSxK(5)       X
C
C        TO CALCULATE SOME MORE INTERMEDIATE VALUES INVOLVED IN
C                COMPUTING THE C-VALUES FOR FREESPACE
C                PROPAGATION
C
C        RETURNS:
C                RVSxI(5)       X
C                RVSxJ(5)       X-ARRAYS OF INTERMEDIATE VALUES
C                RVSxK(5)       X
C
C-----------------------------------------------------------------
C
       REAL*8 RVSxI(5), RVSxJ(5), RVSxK(5), RVCxXI, RVCxYI, RVCxZI
       REAL*8 RVCxLAI, RVCxLOI, RVCxL1, RVCxLO1, RTCxA, RTCxB
       REAL*8 RTCxC, RTCxD
```

Line# Source Line

```
C
          COMMON /START/ RVCxXI, RVCxYI, RVCxZI, RVCxLAI, RVCxLOI
          COMMON /OTHER/ RVCxL1,RVCxLO1,RTCxA,RTCxB,RTCxC,RTCxD
C
          RVSxJ(4) = DSIN(RVCxLAI)
          RVSxJ(5) = DCOS(RVCxLAI)
          RVSxK(4) = DSIN(RVCxLOI - RVCxLO1)
          RVSxK(5) = DCOS(RVCxLOI - RVCxLO1)
          RVSxI(1) = RVSxI(4)*RVSxJ(4)*RVSxK(5) + RVSxI(5)*RVSxJ(5)
          RVSxI(2) = RVSxJ(4)*RVSxK(4)
          RVSxI(3) = RVSxI(5)*RVSxJ(4)*RVSxK(5) - RVSxI(4)*RVSxJ(5
          RVSxJ(1) = -RVSxI(4)*RVSxK(4)
          RVSxJ(2) = RVSxK(5)
          RVSxJ(3) = -RVSxI(5)*RVSxK(4)
          RVSxK(1) = RVSxI(4)*RVSxJ(5)*RVSxK(5) - RVSxI(5)*RVSxJ(4
          RVSxK(2) = RVSxJ(5)*RVSxK(4)
          RVSxK(3) = RVSxI(5)*RVSxJ(5)*RVSxK(5) + RVSxI(4)*RVSxJ(4)
          RETURN
          END
```

FREESP Local Symbols

Name	Class	Type	Size
RVSXK	param		
RVSXJ	param		
RVSXI	param		
RVCXXI.	START	REAL*8	8
RVCXYI.	START	REAL*8	8
RVCXZI.	START	REAL*8	8
RVCXLAI	START	REAL*8	8
RVCXLOI	START	REAL*8	8
RVCXL1.	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	OTHER	REAL*8	8
RTCXD	OTHER	REAL*8	8

Line# Source Line

```
      SUBROUTINE TIMES(RVSxBUN, IVSxNB, RVSxLONO)
C
C-----------------------------------------------------------------
C
C  TIMES -- SUBROUTINE TO CONVERT GIVEN TIME INTO THE TIME
C                VARIABLES NEEDED LATER
C
C       CALLED BY: RAYSUB
C
C-----------------------------------------------------------------
C
C       AUTHOR:          ERIC L. STROBEL
C
C       DATE:            03/18/88
C
C       VERSION:         2.0
C
C-----------------------------------------------------------------
C
C       REVISED:         08/07/86 -- INITIAL REVISION.
C
C                        10/10/86 -- V1.1.  Changed to conform to
C                        the status of RAYTRACE as a subroutine.
C
C                        03/18/88 -- V2.0.  Vastly rewritten to give
C                        a better idea of the mid-hop longitude, and
C                        to better handle being off by factors of
C                        (2 * pi) in the mid-hop longitude.
C
C-----------------------------------------------------------------
C
C      USES:   RVSxBUN(11,15)          ARRAY OF BOUNCE PT. DATA
C              IVSxNB                  BOUNCE COUNT
C              RVSxLONO                LONG. OF LAUNCH PT.
C              ICLxM(11)               NUMBER OF DAYS IN EACH MO.
C              IVCxTIM(5)              DATE AND TIME ARRAY
C
C       TO CALCULATE THE DECIMAL YEAR VALUE AND THE TIME SINCE
C              MIDNIGHT IN RADIANS
C
C       RETURNS:
C              RVCxYR           DECIMAL YEAR
C              RVCxTIM          TIME SINCE MIDNIGHT IN RADIANS
C
C-----------------------------------------------------------------
C
      REAL*8 RVCxYR, RVCxTIM, RVSxBUN(11,15), RPCxPI, RPCxDTOR
      REAL*8 RVCxA, RTLxLON, RVSxLONO, RTCxA, RTCxB, RTLxL0
      REAL*8 RTLxL1, RTLxDEL
C
```

Line#   Source Line

```
          INTEGER ICLxM(11), IVCxTIM(5), ITLxA, ITLxB, IVSxNB
          INTEGER IVCxSSN, IFCxGND, IVLxNR
C
          COMMON /PRAM/ RPCxPI, RPCxDTOR, RTCxA, RTCxB
          COMMON /LPARM/ IVCxSSN, IVCxTIM, RVCxYR, RVCxA, IFCxGND,
     *RVCxTIM
C
          DATA ICLxM/31,28,31,30,31,30,31,31,30,31,30/
C
10000     ITLxA = IVCxTIM(2) - 1
          IVLxNR = 1
          ITLxB = IVCxTIM(3)
          DO 10100 I = 1, ITLxA
              ITLxB = ITLxB + ICLxM(I)
10100     CONTINUE
          RVCxYR = IVCxTIM(1) + ITLxB/365.0D00 - 1980.0D00
          RVCxTIM = IVCxTIM(4) + IVCxTIM(5)/60.0D00
C
C         Compute the longitude of the midpoint of a hop, for the
C           absorption calculation.
C
          IF (IVSxNB.GT.1) THEN
              RTLxL0 = RVSxBUN(IVSxNB-1,2)
          ELSE
              RTLxL0 = RVSxLON0
          ENDIF
          RTLxL1 = RVSxBUN(IVSxNB,2)
          RTLxDEL = RTLxL1 - RTLxL0
          IF (DABS(RTLxDEL).GT.RPCxPI) RTLxDEL = RTLxDEL-INTSIGN(RTLxDEL)
     #*(2.0D00 * RPCxPI)
          RTLxLON = RTLxL0 + RTLxDEL/2.0D00
C
          RVCxTIM = RTLxLON + 15.0D00 * RPCxDTOR * RVCxTIM
10200     IF (RVCxTIM.GE.(2.0D00*RPCxPI)) THEN
              RVCxTIM = RVCxTIM - 2.0D00*RPCxPI
              GO TO 10200
          ELSE IF (RVCxTIM.LT.0.0D00) THEN
              RVCxTIM = RVCxTIM + 2.0D00*RPCxPI
              GO TO 10200
          ENDIF
          RETURN
          END
```

TIMES Local Symbols

Name	Class	Type	Size
RVSXLON0.	param		
IVSXNB.	param		

TIMES Local Symbols

Name	Class	Type	Size
RVSXBUN	param		
ICLXM	local	INTEGER*4	44
ITLXA	local	INTEGER*4	4
ITLXB	local	INTEGER*4	4
RTLXL0.	local	REAL*8	8
RTLXL1.	local	REAL*8	8
I	local	INTEGER*4	4
RTLXDEL	local	REAL*8	8
IVLXNR.	local	INTEGER*4	4
RTLXLON	local	REAL*8	8
RVCXYR.	LPARM	REAL*8	8
RVCXTIM	LPARM	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RPCXDTOR.	PRAM	REAL*8	8
RVCXA	LPARM	REAL*8	8
RTCXA	PRAM	REAL*8	8
RTCXB	PRAM	REAL*8	8
IVCXTIM	LPARM	INTEGER*4	20
IVCXSSN	LPARM	INTEGER*4	4
IFCXGND	LPARM	INTEGER*4	4

Line# Source Line

```
      SUBROUTINE LOSS(RVSxBOU,RVSxBUN,IVSxNB,RVSxLATO,RVSxLONO
     *,RVSxELO,IFSxEND)
C
C-----------------------------------------------------------------
C
C   LOSS -- SUBROUTINE TO EVALUATE THE LOSSES ALONG THE RAY'S
C                 PATH
C
C        CALLED BY: RAYSUB
C
C-----------------------------------------------------------------
C
C        AUTHOR:          ERIC L. STROBEL
C
C        DATE:            09/01/87
C
C        VERSION:         3.0
C
C-----------------------------------------------------------------
C
C        REVISED:         08/07/86 -- INITIAL REVISION.
C
C                         09/05/86 -- V2.0.  ADD HORIZON EFFECTS.
C                         BUG FIX.  COSMETIC CHANGES.
C
C                         10/10/86 -- V2.1.  Changed to conform to
C                         the new status of RAYTRACE as a subrout.
C                         Also trimmed some of the commented out
C                         code.
C
C                         04/09/87 -- V2.2.  Corrected error in
C                         calculation of geometric loss.
C
C                         09/01/87 -- V3.0.  Changed to comment out
C                         absorption loss calculation in anticipation
C                         of a better routine.  Also, the kludgy
C                         "excess loss" has been commented out.  The
C                         focusing calculation has been generalized
C                         and some corrections have been made.
C
C-----------------------------------------------------------------
C
C       USES:   RVSxBOU(11,15)            BOUNCE PT. ARRAY
C               RVSxBUN(4,3,11)           RAY BUNDLE ARRAY
C               IVSxNB                    BOUNCE COUNT
C               RVSxLATO                  LAT OF LAUNCH POINT
C               RVSxLONO                  LON OF LAUNCH POINT
C               IFSxEND                   FLAGS THE LOSS COMP. FIP
C                                            END OF THE RAY
C               RVCxFSQ                   WAVE FREQUENCY SQUARED
```

```
Line#  Source Line

C                IVCxSSN                  SUNSPOT NUMBER
C                IVCxTIM(5)               DATE AND TIME ARRAY
C                RVCxYEAR                 DECIMAL YEAR
C                RVCxA100                 ANGLE OF INCIDENCE AT
C                                            100 KM ALTITUDE
C                IFCxGND                  TYPE OF REFLECTION SURF.
C                RVCxTIM                  TIME SINCE MIDNIGHT IN
C                                            RADIANS
C
C         TO CALCULATE THE SIGNAL LOSS CONTRIBUTIONS FROM:
C                ABSORPTION
C                REFLECTION
C                OTHER PROCESSES
C                GEOMETRIC FOCUSING/DEFOCUSING
C
C         RETURNS:
C                RVCxLOSA                 ABSORPTION LOSS
C                RVCxLOSR                 REFLECTION LOSS
C                RVCxLOSX                 EXCESS LOSS
C                RVCxLOSG                 GEOMETRIC LOSS
C
C-------------------------------------------------------------------
C
      REAL*8 RTLxK1, RVLxMOT, RVCxYEAR, RVLxDEC, RVLxLAT
      REAL*8 RVSxBUN(4,3,11), RVSxLAT0, RVSxLON0, RVLxZEN
      REAL*8 RVLxZ12, RVCxA100, RVLxW, RVLxM, RVLxK, RTLxA
      REAL*8 RTLxB, RTLxC, RTLxDEN, RVCxLOSA, RTLxRE, RTLxIM
      REAL*8 RTLxARG, RVCxLOSR, RCTxA2, RCTxB2, RCTxC2, RVLxELEV
      REAL*8 RCTxA3, RCTxB3, RCTxC3, RTLxGC, RTLxGP, RVLxLON
      REAL*8 RTLxGLA, RTLxGLO, RTLxDLA, RTLxDLO, RVLxMC
      REAL*8 RVLxGML, RVCxLOSX, RCLxAU, RTLxR, RTLxS(3,3)
      REAL*8 RTLxV1(3), RTLxV2(3), RTLxDE, RTLxCT(3),RTLxCV(3)
      REAL*8 RVLxAREA, RTLxT(3), RTLxCRS(3), RTLxMAG1, RTLxMAG2
      REAL*8 RVCxLOSG, RPCxPI, RPCxDTOR, RPCxRE, RVSxBCU(11,15)
      REAL*8 RVCxII, RVCxJJ, RVCxKK, RVCxFSQ, RVLxFRE
      REAL*8 RVCxTIM, RVLxHOR  RVCxCY, RVCxCZ, RVSxEL0
      REAL*8 RVCxSEZTX(3,3), RTCx1, RTCx2, RTCx3, RTCx4, RTCx5
      REAL*8 RTCx6, RTCx7, RTCx8, RVCxHMIN, RVCxH5, RVCxCX
      REAL*8 RTCxA, RTCxB, RTCxC
C
      COMPLEX*16 CTLxNSQ, CTLxCSQ, CTLxSIN, CTLxSQT
      COMPLEX*16 CVLxRH, CVLxRV
C
      INTEGER IVCxSX, IVCxSY, IVCxSZ
      INTEGER IVSxNB, IVCxTIM(5), IFCxGND, IVCxSSN, IFSxEND
C
      COMMON /MISC/ RVCxSEZTX,RTCx1,RTCx2,RTCx3,RTCx4,RTCx5,
     #RVCxHMIN
      COMMON /MORE/ IVCxSX IVCxSY,IVCxSZ,RVCxCX,RVCxCY,RVCxCZ
      COMMON /END/ RTCx6,RTCx7,RTCx8,RVCxH5
```

Line# Source Line

```
          COMMON /LOCAL/ RVLxELEV
          COMMON /PRAM/ RPCxPI, RPCxDTOR, RPCxRE, RTCxC
          COMMON /OTHER/ RVCxII,RVCxJJ,RVCxKK,RVCxFSQ,RTCxA,RTCxB
          COMMON /LPARM/ IVCxSSN,IVCxTIM,RVCxYEAR,RVCxA100,IFCxGND
     *,RVCxTIM
          COMMON /LOSSES/ RVCxLOSA, RVCxLOSR, RVCxLOSX, RVCxLOSG
C
C   ABSORPTION LOSS CALC (ICLxI IS A TEMPORARY VBL. TO DENOTE
C                                  WHICH RAY IS BEING DONE)
C
C10000   IF (IFSxEND.EQ.1.AND.IVSxNB.EQ.1.AND.RVCxHMIN.GT.30.0)
C        #GO TO 20000
C         IF (IFSxEND.EQ.1.AND.IVSxNB.GT.1.AND.RVCxH5.LT.60.0)
C        #GO TO 20000
C         RTLxK1 = 0.0D00
C         RVLxMOT = (( RVCxYEAR - JIDINT(RVCxYEAR))*365.0D00 + 16.0D00)
C        *30.41667D00
C         RVLxDEC = 0.398D00*DSIN(RPCxPI*(RVLxMOT - 3.17D00)/6.0D00)
C         RVLxDEC = DASIN(RVLxDEC)
C         RVLxLAT = ((2.0D00*IVSxNB - 1.0D00)/(2.0D00*IVSxNB)) *
C        *(RVSxBOU(IVSxNB,1) - RVSxLAT0) + RVSxLAT0
C         RVLxZEN = DSIN(RVLxLAT)*DSIN(RVLxDEC) - DCOS(RVLxLAT)*DCOS(RVLx
C        *DEC) * DCOS(RVCxTIM)
C         RVLxZEN = DACOS(RVLxZEN)
C         RVLxZ12 = RVLxLAT - RVLxDEC
C         RVLxDEC = RVLxDEC / RPCxDTOR
C         RVLxLAT = RVLxLAT / RPCxDTOR
C         RVLxZEN = RVLxZEN / RPCxDTOR
C         RVLxZ12 = RVLxZ12 / RPCxDTOR
C         RVCxA100 = RVCxA100 / RPCxDTOR
C         IF (RVLxLAT.GE.30.0D00) THEN
C             IF (IVCxTIM(2).EQ.11) THEN
C                 RTLxK1 = 0.0083D00
C             ELSE IF (IVCxTIM(2).EQ.12) THEN
C                 RTLxK1 = 0.0282D00
C             ELSE IF (IVCxTIM(2).EQ.1) THEN
C                 RTLxK1 = 0.0269D00
C             ELSE IF (IVCxTIM(2).EQ.2) THEN
C                 RTLxK1 = 0.0089D00
C             ENDIF
C         ENDIF
C         RVLxW = 30.0D00 - DABS(RVLxLAT - 60.0D00)
C         RVLxW = 1.0D00 - RTLxK1*RVLxW
C         RVLxM = 2.25D00 - 0.032D00*RVLxLAT
C         IF (RVLxZEN.GT.90.0D00) THEN
C             RVLxK = 0.01D00
C         ELSE
C             RTLxA = DCOSD(0.393D00*RVLxZEN)
C             RTLxB = DCOSD(0.893D00*RVLxZ12)
C             RTLxC = DCOSD(RVLxZ12)
```

Line# Source Line

```
C              RTLxC = RTLxC ** RVLxM
C              RVLxK = (1.0D00 + 0.005D00*IVCxSSN) * RTLxC * RTLxA / RTLxB
C          ENDIF
           RVLxFRE = DSQRT(RVCxFSQ)
C          RTLxDEN = 10.2D00 + (RVLxFRE + 1.4D00)*(RVLxFRE + 1.4D00)
C          RTLxDEN = RTLxDEN * DCOSD(RVCxA100)
C          RVCxLOSA = RVCxLOSA + (286.0D00*(1.0D00+0.0087D00*RVLxLAT)*
C        *RVLxW*RVLxK / RTLxDEN)
C
C    REFLECTION LOSS CALCULATION, a Fresnel coefficient calculation.
C       For details, see the Radio Science paper referred to in the
C       documentation.
C
20000      IF (IVSxNB-1.LE.0) GO TO 40000
           IF (IFCxGND.EQ.1) THEN
               RTLxRE = 7.0D00
               RTLxIM = -0.005D00 * 18000.0D00 / RVLxFRE
               CTLxNSQ = DCMPLX(RTLxRE, RTLxIM)
           ELSE
               RTLxRE = 80.0D00
               RTLxIM = -5.0D00 * 18000.0D00 / RVLxFRE
               CTLxNSQ = DCMPLX(RTLxRE, RTLxIM)
           ENDIF
           CTLxCSQ = DCMPLX(DCOS(RVLxELEV)*DCOS(RVLxELEV))
           CTLxSIN = DCMPLX(DSIN(RVLxELEV))
           CTLxSQT = CDSQRT(CTLxNSQ - CTLxCSQ)
           CVLxRH = (CTLxSIN - CTLxSQT)/(CTLxSIN + CTLxSQT)
           CVLxRV = (CTLxNSQ*CTLxSIN-CTLxSQT)/(CTLxNSQ*CTLxSIN-CTLxSQT
           RTLxARG = (CDABS(CVLxRV)*CDABS(CVLxRV) + CDABS(CVLxRH)*CDABS
         *CVLxRH)) / 2.0D00
           RVCxLOSR = RVCxLOSR + DABS(10.0D00*DLOG10(RTLxARG))
C
C    EXCESS LOSS CALCULATION
C
C30000      RCTxA2 = 11.18959D00
C          RCTxB2 = -0.02916595D00
C          RCTxC2 = -8.5401D-04
C          RCTxA3 = -70.74865D00
C          RCTxB3 = -0.04418982D00
C          RCTxC3 = 0.002169971D00
C          RTLxGC = RCTxA2 + RVCxYEAR*(RCTxB2 + RVCxYEAR*RCTxC2)
C          RTLxGP = RCTxA3 + RVCxYEAR*(RCTxB3 + RVCxYEAR*RCTxC3
C          RVLxLON = ((2.0D00*IVSxNB - 1.0D00)/(2.0D00*IVSxNB)  *
C        *(RVSxBOU(IVSxNB,2) - RVSxLON0) + RVSxLON0
C          RTLxGLA = (90.0D00 - RTLxGC) * RPCxDTOR
C          RTLxGLO = RTLxGP * RPCxDTOR
C          RTLxDLA = RTLxGLA - RVLxLAT*RPCxDTOR
C          RTLxDLO = RTLxGLO - RVLxLON
C          RVLxMC = DCOS(RTLxDLA) * DCOS(RTLxDLO)
C          RVLxMC = DACOS(RVLxMC)
```

Line# Source Line

```
C          RVLxGML = (RPCxPI/2.0D00 - RVLxMC) / RPCxDTOR
C          IF (RVLxGML.GT.60.0D00) THEN
C              RVCxLOSX = 13.3D00
C          ELSE
C              RVCxLOSX = 9.8D00
C          ENDIF
C
C    GEOMETRIC LOSS CALCULATION
C
40000      RCLxAU = 1.903858875D-09*DCOS(RVSxEL0)
C
C          RCLxAU is the cross-sectional area of a bundle of rays
C            bounded by a square defined by azimuth and elevation
C            deviations of 0.0025 degrees from one corner.  This
C            cross-section is taken at a distance of 1 km from the
C            starting point of the problem.  Only three rays are
C            necessary to define the area, the area may be taken
C            as represented by the vector cross product of the
C            vector from the 'corner' point to one of the other
C            points, with the other similarly taken vector.  This
C            is in fact how the calculation is done.
C
           DO 40100 I = 1,3
C
C          RVSxBUN contains the endpoint locations for the rays in
C            the bundle.
C
               RTLxR = RPCxRE - RVSxBUN(I,3,IVSxNB)
C
C          The RTLxS are the GEC coordinate values for the endpoints
C            of the three rays that define the area.
C
               RTLxS(I,1) = RTLxR*DCOS(RVSxBUN(I,1,IVSxNB))*DCOS(RVSxBUN
     #(I,2,IVSxNB))
               RTLxS(I,2) = RTLxR*DCOS(RVSxBUN(I,1,IVSxNB))*DSIN(RVSxBUN
     #(I,2,IVSxNB))
               RTLxS(I,3) = RTLxR*DSIN(RVSxBUN(I,1,IVSxNB))
40100      CONTINUE
C
C          These next two quantities are the vectors (in GEC components
C            representing the deviation of two of the points from the
C            third (the 'corner').
C
           DO 40200 I = 1,3
               RTLxV1(I) = RTLxS(1,I) - RTLxS(2,I)
               RTLxV2(I) = RTLxS(3,I) - RTLxS(2,I)
40200      CONTINUE
           RTLxDE = SQRT(RVCxCX+RVCxCY+RVCxCZ)
C
C          This next vector is the ray direction in GEC components.
```

Line# Source Line

```
      C
            RTLxCT(1) = IVCxSX * DSQRT(RVCxCX) / RTLxDE
            RTLxCT(2) = IVCxSY * DSQRT(RVCxCY) / RTLxDE
            RTLxCT(3) = IVCxSZ * DSQRT(RVCxCZ) / RTLxDE
      C
      C     Next, the ray direction vector is transformed into GEC
      C       coordinates.
      C
            DO 40300 I = 1,3
               RTLxCV(I) = 0.0
               DO 40250 J = 1,3
                  RTLxCV(I) = RTLxCV(I) + RVCxSEZTX(I,J)*RTLxCT(J)
      40250    CONTINUE
      40300 CONTINUE
            RVLxAREA = 0.0
      C
      C     Now, the triple vector product is calculated which will
      C       give the cross-sectional area of the ray bundle in the
      C       plane defined by the ray direction vector.
      C
            DO 40400 I = 1,3
               RTLxT(I) = RTLxV2(MOD(I,3)+1)*RTLxCV(MOD(I+1,3)+1)
          # - RTLxV2(MOD(I+1,3)+1)*RTLxCV(MOD(I,3)+1)
               RVLxAREA = RVLxAREA + RTLxT(I)*RTLxV1(I)
      40400 CONTINUE
      C
      C     A calculation now begins to obtain the angle between
      C       the ray direction and the surface determined by the
      C       cutoff / earth bounce condition.
      C
            RVLxAREA = DABS(RVLxAREA)
            RTLxMAG1 = 0.0
            RTLxMAG2 = 0.0
            DO 40500 I = 1,3
               RTLxCRS(I) = RTLxV1(MOD(I,3)+1)*RTLxV2(MOD(I+1,3)+1
          # - RTLxV1(MOD(I+1,3)+1)*RTLxV2(MOD(I,3)+1)
               RTLxMAG1 = RTLxMAG1 + RTLxCRS(I)*RTLxCRS(I)
               RTLxMAG2 = RTLxMAG2 + RTLxCV(I)*RTLxCV(I)
      40500 CONTINUE
            RTLxMAG1 = DSQRT(RTLxMAG1)
            RTLxMAG2 = DSQRT(RTLxMAG2)
            RVLxELEV = RVLxAREA / (RTLxMAG1*RTLxMAG2)
            RVLxELEV = DACOS(RVLxELEV)
            RVLxELEV = (RPCxPI/2.0) - RVLxELEV
      C
      C     The ordinary geometric loss calculation.
      C
            RVCxLOSG = 60.0D00 + 10.0D00 * DLOG10(RVLxAREA/RCLxAU
      C
      C     If a ray came into the earth at a sufficiently small angle
```

```
Line#   Source Line

        C          then the empirical horizon focusing calculation is done.
        C          The result of this is then compared to the normal focusing
        C          result and the greater loss is used.
        C
                 IF (ABS(RVLxELEV/RPCxDTOR).LT.1.0D00) THEN
                    RVLxHOR = 60.012D00 + 20.0D00*DLOG10(DBLE(IVSxNB))/3.0D00
             *-10.0D00*DLOG10(DCOS(RVLxELEV)) + 10.0D00*DLOG10(DSIN(RVSxBOU(
             *IVSxNB,4))) - 10.0D00*DLOG10(RVLxFRE)/3.0D00
                    IF (RVLxHOR.GT.RVCxLOSG) RVCxLOSG = RVLxHOR
                 ENDIF
                 RETURN
                 END
```

LOSS Local Symbols

Name	Class	Type	Size
IFSXEND	param		
RVSXEL0	param		
RVSXLON0.	param		
RVSXLAT0.	param		
IVSXNB.	param		
RVSXBUN	param		
RVSXBOU	param		
I	local	INTEGER*4	4
J	local	INTEGER*4	4
RCLXAU.	local	REAL*8	8
RTLXMAG1.	local	REAL*8	8
RTLXMAG2.	local	REAL*8	8
RTLXV1.	local	REAL*8	24
RTLXV2.	local	REAL*8	24
RTLXDE.	local	REAL*8	8
CVLXRH.	local	COMPLEX*16	16
RTLXR	local	REAL*8	8
RTLXS	local	REAL*8	72
RTLXT	local	REAL*8	24
RTLXIM.	local	REAL*8	8
RTLXCT.	local	REAL*8	24
RTLXRE.	local	REAL*8	8
CTLXCSQ	local	COMPLEX*16	16
RTLXCV.	local	REAL*8	24
RTLXARG	local	REAL*8	8
CVLXRV.	local	COMPLEX*16	16
CTLXSIN	local	COMPLEX*16	16
RVLXAREA.	local	REAL*8	8
RVLXFRE	local	REAL*8	8
CTLXNSQ	local	COMPLEX*16	16
RTLXCRS	local	REAL*8	24
CTLXSQT	local	COMPLEX*16	16

LOSS Local Symbols

Name	Class	Type	Size
RVLXHOR	local	REAL*8	8
RVCXLOSG	LOSSES	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXDTOR	PRAM	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RVCXII	OTHER	REAL*8	8
RVCXJJ	OTHER	REAL*8	8
RVCXKK	OTHER	REAL*8	8
RVCXFSQ	OTHER	REAL*8	8
RVCXTIM	LPARM	REAL*8	8
RVCXCY	MORE	REAL*8	8
RVCXCZ	MORE	REAL*8	8
RVCXSEZTX	MISC	REAL*8	72
RTCX1	MISC	REAL*8	8
RTCX2	MISC	REAL*8	8
RVCXYEAR	LPARM	REAL*8	8
RTCX3	MISC	REAL*8	8
RTCX4	MISC	REAL*8	8
RTCX5	MISC	REAL*8	8
RTCX6	END	REAL*8	8
RTCX7	END	REAL*8	8
RTCX8	END	REAL*8	8
RVCXA100	LPARM	REAL*8	8
RVCXHMIN	MISC	REAL*8	8
RVCXH5	END	REAL*8	8
RVCXCX	MORE	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	PRAM	REAL*8	8
RVCXLOSA	LOSSES	REAL*8	8
RVCXLOSR	LOSSES	REAL*8	8
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
IVCXSZ	MORE	INTEGER*4	4
RVLXELEV	LOCAL	REAL*8	8
IVCXTIM	LPARM	INTEGER*4	20
IFCXGND	LPARM	INTEGER*4	4
IVCXSSN	LPARM	INTEGER*4	4
RVCXLOSX	LOSSES	REAL*8	8

Line# Source Line

```
      SUBROUTINE GCDEV(IVSxNB,RVSxAZ,RVSxLA0,RVSxLO0,RVSxBOU,RVSxDEV)
C
C----------------------------------------------------------------------
C
C   GCDEV -- SUBROUTINE TO EVALUATE THE DEVIATION OF A RAY FROM
C                  A GREAT CIRCLE PATH
C
C        CALLED BY: RAYSUB
C
C----------------------------------------------------------------------
C
C        AUTHOR:          ERIC L. STROBEL
C
C        DATE:            10/10/86
C
C        VERSION:         2.1
C
C----------------------------------------------------------------------
C
C        REVISED:         09/05/86 -- INITIAL REVISION.
C
C                         09/17/86 -- V2.0.  Practically a complete
C                         rewrite.  Now does the spherical trig OK
C                         even in the special cases... I think.
C
C                         10/10/86 -- V2.1.  Changed to conform to
C                         the new status of RAYTRACE as a subrout.
C
C----------------------------------------------------------------------
C
C       USES:   RVSxBOU(11,15)          BOUNCE PT. ARRAY
C               IVSxNB                  BOUNCE COUNT
C               RVSxAZ                  LAUNCH AZIMUTH
C               RVSxLA0                 LAUNCH LATITUDE
C               RVSxLO0                 LAUNCH LONGITUDE
C
C        TO CALCULATE THE DEVIATION OF THE RAYPATH AWAY FROM ITS
C                EXPECTED LANDING POINT (IF IT HAD FOLLOWED A
C                GREAT CIRCLE PATH).
C
C        RETURNS:
C               RVSxDEV                 DEVIATION IN KM
C
C----------------------------------------------------------------------
C
      INTEGER IVSxNB, ITLxS, ITLxC
C
      REAL*8 RVSxDEV, RVSxBOU(11,15), RVSxAZ
      REAL*8 RVSxLA0, RVSxLO0, RPCxPI, RPCxRE, RPCxDTR
      REAL*8 RVLxR, RTLx1, RVLxLAP, RVLxLOP, RVLxSDL
```

```
Line#  Source Line

              REAL*8 RVLxCDL, RVLxDL, RTLxA, RTLxB, RTLxD, RTCxA
      C
              COMMON /PRAM/ RPCxPI, RPCxDTR, RPCxRE, RTCxA
      C
      C       First, do the spherical trig to get the great circle
      C         landing point (i.e., where the ray would have landed,
      C         given the range that it went, if it had stayed on the
      C         great circle path at the original azimuth).
      C
              RVLxR = RVSxBOU(IVSxNB, 4)
              RTLx1 = DSIN(RVSxLA0)*DCOS(RVLxR) + DCOS(RVSxLA0)*
           *DSIN(RVLxR)*DCOS(RVSxAZ)
              RVLxLAP = DASIN(RTLx1)
              RVLxSDL = DSIN(RVSxAZ)*DSIN(RVLxR)/DCOS(RVLxLAP)
              RVLxCDL = (DCOS(RVLxR)-DSIN(RVSxLA0)*DSIN(RVLxLAP))/
           *(DCOS(RVSxLA0)*DCOS(RVLxLAP))
              RVLxDL = DASIN(RVLxSDL)
              ITLxS = INTSIGN(RVLxSDL)
              ITLxC = INTSIGN(RVLxCDL)
              IF (ITLxC.LT.0) RVLxDL = ITLxS*RPCxPI - RVLxDL
              IF (ITLxS.EQ.0.AND.ITLxC.LT.0) RVLxDL = RPCxPI
              RVLxLOP = RVSxLO0 + RVLxDL
      C
      C       Now, calculate the great circle distance between the actual
      C         landing point and the great circle landing point.
      C
              RTLxA = RVSxBOU(IVSxNB, 1)
              RTLxB = RVSxBOU(IVSxNB, 2)
              RTLxD = RTLxB - RVLxLOP
              RVSxDEV = DSIN(RTLxA)*DSIN(RVLxLAP) + DCOS(RTLxA)*
           #DCOS(RVLxLAP)*DCOS(RTLxD)
              IF ((RVSxDEV - 1.0D00).GT.0.0D00) RVSxDEV = 1.0D00
              IF ((RVSxDEV + 1.0D00).LT.0.0D00) RVSxDEV = -1.0D00
              RVSxDEV = RPCxRE*DACOS(RVSxDEV)
              RETURN
              END
```

GCDEV Local Symbols

Name	Class	Type	Size
RVSXDEV	param		
RVSXBOU	param		
RVSXLO0	param		
RVSXLA0	param		
RVSXAZ.	param		
IVSXNB.	param		
ITLXC	local	INTEGER*4	4
RTLXA	local	REAL*8	8

GCDEV Local Symbols

Name	Class	Type	Size
RTLXB	local	REAL*8	8
RTLXD	local	REAL*8	8
ITLXS	local	INTEGER*4	4
RVLXDL	local	REAL*8	8
RVLXR	local	REAL*8	8
RVLXCDL	local	REAL*8	8
RVLXLAP	local	REAL*8	8
RVLXSDL	local	REAL*8	8
RVLXLOP	local	REAL*8	8
RTLX1	local	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RPCXDTR	PRAM	REAL*8	8
RTCXA	PRAM	REAL*8	8

Line# Source Line

```
      SUBROUTINE ANRANG(RVSxAZ, RVSxLA1, RVSxLO1, RVSxLA0, RVSxLO0
     #, RVSxR)
C
C-----------------------------------------------------------------
C
C     ANRANG  -- SUBROUTINE TO COMPUTE THE ANGULAR RANGE BETWEEN
C                TWO POINTS
C
C
C     CALLED BY:  RAYSUB
C
C-----------------------------------------------------------------
C
C     AUTHOR:            ERIC L. STROBEL
C
C     DATE:              03/18/88
C
C     VERSION:           2.0
C
C-----------------------------------------------------------------
C
C     REVISED:           05/01/87 -- V1.0.  Initial revision.
C
C                        03/18/88 -- V2.0.  Rewritten to handle using
C                        the correct azimuth between the launch point
C                        and the current point, as opposed to using the
C                        launch azimuth.  Also, now correctly
C                        handles azimuths of 0 & 180 degrees, as
C                        well as angular ranges > 180 degrees.
C
C-----------------------------------------------------------------
C
C     USES:              RVSxAZ       Azimuth.
C
C                        RVSxLA1 \
C                        RVSxLO1 |_   The endpoints of the path.
C                        RVSxLA0 |
C                        RVSxLO0 /
C
C
C     To calculate the angular range along the great circle path
C       between the two points.
C
C
C     RETURNS:           RVSxR        The angular range.
C
C-----------------------------------------------------------------
C
      REAL*8 RTLx1, RVSxAZ, RVSxLA1, RVSxLO1, RVSxLA0, RVLxTH0
      REAL*8 RVSxLO0, RVSxR, RVLxCR, RPCxPI, RVLxPIO2, RVLxTH1
```

Line# Source Line

```
            REAL*8 RTCxA, RTCxB, RTCxC, RVLxS0, RVLxC0, RVLxS1, RVLxC1
            REAL*8 RVLxS2, RVLxC2, RVLxTAD, RVLxAC1, RVLxAC2, RVLxTAN
            REAL*8 RVLxAC, RVLxD1, RVLxD2
      C
            INTEGER IVLxSS, IVLxSC
      C
            COMMON /PRAM/ RPCxPI, RTCxA, RTCxB, RTCxC
      C
            RVLxPIO2 = RPCxPI/2.0D00
            RTLx1 = RVSxLO1 - RVSxLO0
            RVLxTH0 = RVLxPIO2 - RVSxLA0
            RVLxTH1 = RVLxPIO2 - RVSxLA1
      C
            RVLxS0 = DSIN(RVLxTH0)
            RVLxC0 = DCOS(RVLxTH0)
            RVLxS1 = DSIN(RVLxTH1)
            RVLxC1 = DCOS(RVLxTH1)
            RVLxS2 = DSIN(RTLx1)
            RVLxC2 = DCOS(RTLx1)
      C
            RVLxCR = RVLxC2*RVLxS0*RVLxS1 + RVLxC0*RVLxC1
            RVLxTAD = RVLxC1 - RVLxCR*RVLxC0
      C
      C     We're trying to push spherical trig to yield correct values
      C       even when the sides of the triangle are greater than 180 dec.
      C       Here, the correct azimuth between the endpoints of the range
      C       is being calculated.  In this way, if the range is greater
      C       than 180 degrees, we can get enc..n information to determine
      C       the correct value.
      C
            IF (RVLxTAD.EQ.0.0) THEN
                RVLxAC1 = RVLxPIO2
                RVLxAC2 = 3.0 * RVLxPIO2
            ELSE IF (RVLxS2.EQ.0.0) THEN
                RVLxAC1 = 0.0
                RVLxAC2 = RPCxPI
            ELSE
                RVLxTAN = RVLxS0*RVLxS1*RVLxS2
                RVLxAC = DATAN(RVLxTAN/RVLxTAD)
                IF (RVLxAC.LT.0.0) RVLxAC = RVLxAC + RPCxPI
                RVLxAC1 = RVLxAC
                RVLxAC2 = RVLxAC + RPCxPI
            ENDIF
            RVLxD1 = DABS(RVSxAZ - RVLxAC1)
            RVLxD2 = DABS(RVSxAZ - RVLxAC2)
            IF (RVLxD1.LT.RVLxD2) THEN
                RVSxAZ = RVLxAC1
            ELSE
                RVSxAZ = RVLxAC2
            ENDIF
```

```
Line#   Source Line

      C
      C       Now calculate the angular range.
      C
              IF (RVSxAZ.EQ.0.0.OR.RVSxAZ.EQ.RPCxPI) THEN
                  RVSxR = (RVLxTH0 - RVLxC2*RVLxTH1)*DCOS(RVSxAZ)
              ELSE
                  IVLxSS = INTSIGN(RVLxS2*RVLxS1/DSIN(RVSxAZ))
                  RVSxR = DACOS(RVLxCR)
              ENDIF
      C
      C       Range will be negative if greater than 180 degrees.
      C         Therefore the following line will give the desired
      C         value.
      C
              IF (RVSxR.LT.0.0) RVSxR = 2.0D00*RPCxPI + RVSxR
      C
              RETURN
              END
```

ANRANG Local Symbols

Name	Class	Type	Size
RVSXR	param		
RVSXLO0	param		
RVSXLA0	param		
RVSXLO1	param		
RVSXLA1	param		
RVSXAZ.	param		
RVLXC1.	local	REAL*8	8
RVLXD1.	local	REAL*8	8
RVLXAC1	local	REAL*8	8
RVLXC2.	local	REAL*8	8
RVLXD2.	local	REAL*8	8
RVLXAC2	local	REAL*8	8
RVLXS0.	local	REAL*8	8
RVLXAC.	local	REAL*8	8
RVLXS1.	local	REAL*8	8
RVLXS2.	local	REAL*8	8
RVLXTH0	local	REAL*8	8
RVLXTH1	local	REAL*8	8
RVLXCR.	local	REAL*8	8
RVLXTAD	local	REAL*8	8
RVLXPIO2.	local	REAL*8	8
IVLXSS.	local	INTEGER*4	4
RVLXTAN	local	REAL*8	8
RTLX1	local	REAL*8	8
RVLXC0.	local	REAL*8	8
RPCXPI.	PRAM	REAL*8	8

ANRANG Local Symbols

Name	Class	Type	Size
RTCXA	PRAM	REAL*8	8
RTCXB	PRAM	REAL*8	8
RTCXC	PRAM	REAL*8	8

Line# Source Line

```
      SUBROUTINE INBOX(RVSxLAI,RVSxLOI,RVSxLAS,RVSxLOS
     #,RVSxLAE,RVSxLOE,IVSxOUT)
C
C-----------------------------------------------------------------
C
C      INBOX  -- SUBROUTINE TO DETERMINE WHETHER A GIVEN PT. IS
C                WITHIN THE IONOSPHERIC SPECIFICATION GRID.
C
C
C      CALLED BY:       IONOPAR
C
C-----------------------------------------------------------------
C
C      AUTHOR:          ERIC L. STROBEL
C
C      DATE:            03/01/87
C
C      VERSION:         1.0
C
C-----------------------------------------------------------------
C
C      REVISED:         03/01/87 -- V1.0.  Initial revision.
C
C-----------------------------------------------------------------
C
C      USES:            RVSxLAI, LOI     The coord. of the pt. of
C                                       interest.
C
C                       RVSxLAS \
C                       RVSxLOS |__      The starting and ending
C                       RVSxLAE |       (SW & NE) corners of the
C                       RVSxLOE /       specification grid.
C
C
C      To determine whether or not the point of interest lies
C        within the grid over which the ionosphere is specified.
C        If it is outside, then the program will use a spherically
C        symmetric ionosphere having the parameters of the nearest
C        grid point, otherwise the program will do the interpolation.
C
C
C      RETURNS:         IVSxOUT          A flag specifying whether
C                                       the pt. of interest is in or
C                                       out of the grid. (0 = in,
C                                       1 = out)
C
C-----------------------------------------------------------------
C
       REAL*8 RVSxLAI, RVSxLOI, RVSxLAS, RVSxLOS
       REAL*8 RVSxLAE, RVSxLOE
```

Line# Source Line

```
         REAL*8 RTLx1, RTLx2, RTLx3, RTLx4
 C
         INTEGER IVSxOUT
 C
 C
         RTLx1 = MIN(RVSxLOS,RVSxLOE)
         RTLx2 = MAX(RVSxLOS,RVSxLOE)
         RTLx3 = MIN(RVSxLAS,RVSxLAE)
         RTLx4 = MAX(RVSxLAS,RVSxLAE)
         IF ((((RVSxLAI-RTLx4).LE.0.0D00).AND.
     #((RVSxLAI-RTLx3).GE.0.0D00)).AND.
     #(((RVSxLOI-RTLx2).LE.0.0D00).AND.
     #((RVSxLOI-RTLx1).GE.0.0D00))) THEN
            IVSxOUT = 0
         ELSE
            IVSxOUT = 1
         ENDIF
         RETURN
         END
```

INBOX Local Symbols

Name	Class	Type	Size
IVSXOUT	param		
RVSXLOE	param		
RVSXLAE	param		
RVSXLOS	param		
RVSXLAS	param		
RVSXLOI	param		
RVSXLAI	param		
RTLX1	local	REAL*8	8
RTLX2	local	REAL*8	8
RTLX3	local	REAL*8	8
RTLX4	local	REAL*8	8

Line# Source Line

```
       FUNCTION INTSIGN(R)
C
C       GIVES 1,-1,OR 0 AS THE SIGN OF A REAL*8 NUMBER.  This
C           mimics the SIGN function available in most BASICs.
C
       REAL*8 R
C
       INTEGER INTSIGN
C
       IF (R.EQ.0.0D00) THEN
           INTSIGN = 0
           RETURN
       ENDIF
       INTSIGN = IDNINT(R/DABS(R))
       RETURN
       END
```

INTSIGN Local Symbols

Name	Class	Type	Size
R	param		
INTSIGN	param		

Global Symbols

Name	Class	Type	Size
ACCFSP.	FSUBRT	***	***
ANRANG.	FSUBRT	***	***
END	common	***	32
FREESP.	FSUBRT	***	***
GCDEV	FSUBRT	***	***
GORP.	common	***	16
INBOX	FSUBRT	***	***
INTSIGN	FFUNCT	INTEGER*4	***
IONO1	common	***	32
LOCAL	common	***	8
LOSS.	FSUBRT	***	***
LOSSES.	common	***	32
LPARM	common	***	52
MAINDAT	common	***	48
MISC.	common	***	120
MORE.	common	***	36
NEWCS	FSUBRT	***	***
OTHER	common	***	48
PRAM.	common	***	32
START	common	***	40

```
Global Symbols

Name                          Class    Type              Size

TEMP2 . . . . . . . . . . . . common   ***                 16
TIMES . . . . . . . . . . . . FSUBRT   ***                ***
TRIANG. . . . . . . . . . . . FSUBRT   ***                ***

Code size = 1f47 (8007)
Data size = 0116 (278)
```